Numerical Solution of Differential-Algebraic Systems Arising in Circuit Simulation

vorgelegt von

Dipl.-Math.oec. Simone Bächle

von der Fakultät II - Mathematik und Naturwissenschaften
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
- Dr. rer. nat. -

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr. Rolf Möhring
Berichter:	Prof. Dr. Volker Mehrmann
Berichter:	Prof. Dr. Matthias Bollhöfer
Berichter:	Prof. Dr. Caren Tischendorf

Tag der wissenschaftlichen Aussprache: 21.03.2007

Berlin 2007
D 83

Acknowledgment

During the work on this thesis, many people have lent advice, support and encouragement. It is at this point that I would like to thank these people.

Foremost, I would like to express my gratitude to Prof. Dr. Volker Mehrmann who supervised my work for his patience, support, advice and criticism but also for the freedom to choose my own approach to the subject. My thanks also go to Prof. Dr. Matthias Bollhöfer for interesting discussions on linear iterative solvers even though this aspect dropped out of the scope of this thesis in the end. I would like to thank Prof. Dr. Caren Tischendorf for valuable discussions on circuit simulation and the concept of the tractability index.

My gratitude also goes to Falk Ebert for a most fruitful cooperation and the right idea at the right time, but also for encouraging me whenever I needed encouragement. Another person that I would like to thank is Dr. Andreas Steinbrecher who helped by answering many of my questions concerning the theory of DAEs and the strangeness index. His drawer always seemed to contain just the article I needed to continue my work. Both Falk Ebert and Dr. Andreas Steinbrecher helped me by proofreading this thesis.

As a student worker in the project in which the theory presented in this thesis was developed Eva Abram implemented the graph theoretical part of the algorithms. It was a pleasure to work with her. In regard of the graph theoretical aspects of my work, I also have to thank Dr. Christian Liebchen and Dr. Ekkehard Köhler. With their help, I was able to make some of the existing algorithms more efficient.

I'm grateful that I was able to spent my time as PhD student in a work group that provided such a great atmosphere. I would like to thank all my colleagues for this. Especial thanks go to Sonja Schlauch who had to share an office room with me.

I'm also grateful that the DFG research center Matheon financed the project in which the thesis was created. Moreover, the diversity in aspects of Applied Mathematics which can be found in Matheon offered the unique opportunity to look beyond my own nose.

Last but surely not least I would like to thank my family. I'm grateful to have parents that supported and encouraged me whenever possible and who offered me the possibility to study Mathematics. I thank my husband Steffen Bauer from the bottom of my heart for being there for me whenever I needed someone to comfort and encourage me if things just did not work or to share my happiness if another step in the right direction was taken.

Zusammenfassung

In der vorliegenden Dissertationsschrift werden quasi-lineare Differential-Algebraische Gleichungen (DAE), wie sie in der Schaltungssimulation auftreten, untersucht. Dabei wird von einer Beschreibung der Schaltung als Netzliste, d.h. als Liste aller in der Schaltung enthaltener Bauteile sowie deren Verschaltung, ausgegangen. An Hand dieser Netzliste wird dann mit Hilfe der klassischen oder der ladungsorientierten Modifizierten Knotenanalyse eine DAE gewonnen. Ein Schwerpunkt der Arbeit liegt auf der graphentheoretischen Struktur der Schaltung, die sich in den Eigenschaften der Schaltungs-DAE widerspiegelt. So ist zum Beispiel bekannt, dass sich der Differentationsindex einer solchen DAE unter bestimmten Voraussetzungen aus dem Netzwerkgraphen der Schaltung bestimmen läßt. Weiterhin ist bekannt, dass sich im Zuge dieser graphentheoretischen Indexbestimmung ebenso die Gleichungen bestimmen lassen, aus deren Ableitungen sich versteckte Zwangsbedingungen ergeben.

Im Rahmen dieser Arbeit wurden zwei Verfahren entwickelt, die diese strukturellen Informationen nutzen, um den Differentationsindex zu reduzieren, falls die betrachtete Schaltungs-DAE Differentationsindex 2 hat. Dazu werden die graphentheoretischen Grundlagen erarbeitet und auf bestehende Ergebnisse angewandt. Es wird gezeigt, dass auf Grund dieser graphentheoretischen Grundlagen beide Verfahren zur Indexreduktion ohne rechenaufwendige Rangbestimmungen auskommen und somit auch für größere Schaltungen geeignet sind. Zu dem läßt das zweite Indexreduktionsverfahren, das für DAEs aus der ladungsorientierten Modifizierten Knotenanalyse entwickelt wurde, eine physikalische Deutung als Modifikation der zu Grunde liegenden Schaltung zu. Daher kann dieses Indexreduktionsverfahren nicht nur als rein numerisches Verfahren, das die bestehende Schaltungs-DAE verändert, sonder auch als Verfahren, welches die der DAE zu Grunde liegende Schaltung verändert, realisiert werden kann. Dieses Vorgehen hat den Vorteil, dass bestehende Simulationssoftware nur wenig angepasst werden muss.

Da in der Industrie die ladungsorientierte Modifizierte Knotenanlyse gegenüber der klassischen Variante bevorzugt wird, wurde der akademische Schaltungssimulator PSIM von Robert Melville so modifiziert, dass die ladungsorientiere Formulierung zur Modellierung der Schaltung verwendet wird. Danach wird der Differentationsindex der erzeugten DAE mit Hilfe graphentheoretischer Methoden bestimmt und anschliessend die in dieser Arbeit vorgestellte Indexreduktion für diese Art von DAEs durchgeführt. Tests anhand von einigen Beispielen zeigen die Effizienz und Robustheit dieser Methode im Vergleich zur Simulation der Schaltungen ohne Indexreduktion.

Abstract

In this thesis, quasi-linear Differential-Algebraic Equations (DAE) as they arise in circuit simulation are examined. The circuit is described with help of a netlist, e.g. a list which contains the information about all devices in the circuit and the way these devices are connected to each other. With the help of this netlist a DAE is created by using either the classical or the charge oriented Modified Nodal Analysis. One focus of this thesis is the graph theoretical structure of the circuit, which is reflected in the properties of the circuit DAE. For example, it is known that it is possible under certain conditions to determine the differentiation index of such a DAE based on the graph of the circuit. Moreover, it is possible to determine the derivatives of equations which yield hidden constraints.

In the course of this work, two methods have been developed that use this structural information to reduce the differentiation index of a circuit DAE if the DAE has differentiation index 2. To this end the graph theoretical foundations are laid out and applied to the existing results. It is shown that because of the graph theoretical results both index reduction methods work without time consuming rank decisions. Therefore these methods are applicable to large circuits. Moreover, the second method which was developed for DAEs arising from the charge oriented Modified Nodal Analysis allows for a physical interpretation. Hence this method not only can be realized as a numerical method that alters the DAE in order to reduce the differentiation index, but also as a method that alters the circuit itself. The second approach has the advantage that existing software for circuit simulation only has to be adapted slightly to use the index reduction method.

Since the charge oriented Modified Nodal Analysis is preferred in industrial circuit simulation, the academical circuit simulator Psim by Robert Melville has been adapted to use this variant to produce the circuit DAE. Following this step the differentiation index of the resulting DAE is determined by graph theoretical algorithms and the index reduction method for DAEs from charge oriented Modified Nodal Analysis as proposed in this thesis is applied. The method has been tested on various examples. A comparison with the numerical results obtained without reducing the differentiation index shows the robustness and efficiency of the method.

Contents

List of Figures

List of Tables

List of Algorithms

Notation

$\mathbf{\Phi}$	vector of inductive fluxes, 58
ν_d	differentiation index of a DAE, 25
ν_s	strangeness index of a DAE, 28
ν_t	tractability index of a DAE, 27
$\mathbf{i}$	vector of branch currents, 57
$\mathbf{i}_C$	vector of branch currents through capacitances, 57
$\mathbf{i}_L$	vector of branch currents through inductances, 57
$\mathbf{i}_R$	vector of branch currents through resistances, 57
$\mathbf{i}_V$	vector of branch currents through voltage sources, 57
$\mathbf{q}$	vector of capacitive charges, 58
$\mathbf{v}$	vector of branch voltages, 57
$\mathbf{v}_C$	vector of branch voltages across capacitances, 57
$\mathbf{A}_C$	incidence matrix with respect to capacitances, 57
$\mathbf{A}_I$	incidence matrix with respect to current sources, 57
$\mathbf{A}_L$	incidence matrix with respect to inductances, 57
$\mathbf{A}_R$	incidence matrix with respect to resistances, 57
$\mathbf{A}_V$	incidence matrix with respect to voltage sources, 57
$\mathbf{M}_{CV}$	fundamental CV loop matrix, 83
$\mathbf{S}_{LI}$	fundamental LI cutset matrix, 64
$\bar{\mathbf{Z}}_{V-C}$	matrix with columns forming a basis of $\ker \mathbf{Z}_C^T \mathbf{A}_V$, 60
$\mathbf{Z}_C$	matrix with columns forming a basis of $\ker \mathbf{A}_C^T$, 60
$\mathbf{Z}_{CRV}$	the product $\mathbf{Z}_C \mathbf{Z}_{V-C} \mathbf{Z}_{R-CV}$, 60
$\mathbf{Z}_{R-CV}$	matrix with columns forming a basis of $\ker \mathbf{A}_R^T \mathbf{Z}_C \mathbf{Z}_{V-C}$, 60
$\mathbf{Z}_{V-C}$	matrix with columns forming a basis of $\ker \mathbf{A}_V^T \mathbf{Z}_C$, 60

1. Introduction

Many modern industries like automobile or telecommunication industry depend on the development of new electronic chips. One important step in the design of a VLSI chip is the simulation of its transient behavior. Since the corresponding circuits usually contain several millions of elements, structured approaches are necessary to set up the equations that describe the behaviour of the circuits. The classical and the charge-oriented Modified Nodal Analysis are two such approaches which are widely used in industrial circuit simulation. Both methods combine ordinary differential equations, which model the dynamics of a circuit, with nonlinear algebraic equations, which arise from the Kirchhoff's Laws as well as from the characteristic equations of certain elements in the circuit. Such mixed systems of differential and algebraic equations are referred to as differential-algebraic equations or DAEs.

Although there are well-developed analytical and numerical theories for both ordinary differential equations and nonlinear algebraic equations, the analytical and numerical treatment of DAEs causes problems. For example, initial values for a given DAE may not be chosen arbitrarily since they have to fulfill the algebraic equations. This is quite different from the case of ordinary differential equations where the initial values can be chosen arbitrarily. Moreover, even if initial values which fulfill the algebraic equations are given for a DAE, a solution of the initial value problem does not need to exists or if a solution exists, then it may not be unique. Again, this differs from the behavior of ordinary differential equations where the choice of the initial values usually fixes a unique solution.

Another problem consists in finding a numerical solution for a given uniquely solvable DAE initial value problem. For ordinary differential equations there exists a plethora of numerical methods which are adapted to the various kinds of ordinary differential equations. Of these methods those that are suited for the treatment of stiff ordinary differential equations often have been proven to be applicable for the numerical treatment of DAEs. However, there are many cases in which these methods fail to compute a numerical solution of a uniquely solvable DAE initial value problem.

In order to classify DAEs, various so-called index concepts have been defined. All of these index concepts try to measure in one way or another, how far the behavior of a DAE is from the behavior of a related ordinary differential equation. A well-known index is the so-called differentiation index. In terms of the differentiation index, index 1 DAEs are close to ordinary differential equations. It is known how to find suitable initial values for DAEs with differentiation index 1 and there are results concerning the convergence of those discretization methods which are used for DAEs. For DAEs with differentiation index higher than 1, such results may not exist.

With these difficulties in mind, attempts have been made to reduce the differentiation index of a DAE to 1 in order to be able to apply the existing theory of DAEs with differentiation index 1 [15, 30, 46, 48, 50, 55, 64]. For general nonlinear DAEs, these index reduction methods usually either lead to an overdetermined system or involve several rank decisions which make those methods prohibitively expensive for large DAEs.

For DAEs that arise from circuit simulation, there are results that show that under certain conditions the differentiation index of the DAEs is ≤ 2. Moreover, these results allow for a determination of the differentiation index and those equations, which are needed to reduce the differentiation index, based on the graph of the circuit. Up to now, this information has been used to determine consistent initial values for the simulation of the transient behavior of a given circuit [26, 28, 29]. In this thesis, index reduction methods are proposed that are tailored to DAEs arising from classical and charge-oriented Modified Nodal Analysis. Since the DAEs arising in industrial circuit simulation consist of several millions of equations and unknowns, index reduction methods that use methods like singular value decomposition or the QR algorithm to perform rank decisions are too costly. The same holds for index reduction methods that attempt to solve overdetermined systems. The index reduction methods proposed in this thesis however are based on the information contained in the graph of the circuit. This allows to determine the equations which are needed in order to reduce the index efficiently. Moreover, the structural information also can be used to transform the original DAE in such a way that a DAE with differentiation index 1 results which has the same properties as the original DAE. In particular, the resulting DAE can be interpreted as a circuit DAE again. It can be shown that the efficiency of standard discretization methods for DAEs can be increased if the proposed index reduction method is applied to a given circuit DAE with differentiation index 2.

This thesis is organized as follows. In Chapter 2 we will give a short introduction into the theory of nonlinear DAEs. Section 2.1 contains the basic definitions which are needed later on. Section 2.2 presents three index concepts which are important in the field of DAEs from circuit simulation. Two numerical integration methods which are suited for the treatment of DAEs are introduced in Section 2.3. Finally, an overview of index reduction methods for general nonlinear DAEs is given. Of these index reduction methods the index reduction by minimal extension [48] plays an important role in the development of the index reduction methods for DAEs from circuit simulation presented in this thesis. Section 2.5 gives an overview over existing software for the numerical solution of DAEs.

Chapter 3 focuses on DAEs that arise from classical or charge-oriented Modified Nodal Analysis. Since both variants of the Modified Nodal Analysis are strongly based on the graph that represents a circuit, Section 3.1 introduces the graph theoretical background. This background are not only needed to derive a DAE that describes a circuit in Section 3.2, but they also play an important role in the index reduction methods proposed in this thesis. As another important ingredient of the index reduction methods, the results from [28, 29] concerning the determination of the differentiation index of a circuit DAE and the computation of derivatives needed for the index reduction are summarized in Section 3.3.

In Chapter 4, two methods to reduce the differentiation index of a circuit DAE with differentiation index 2 are proposed. Section 4.1 examines the influence of controlled sources on the proposed index reduction methods. The first index reduction method is presented in Section 4.2. It is tailored to DAEs arising from the classical Modified Nodal Analysis. The second index reduction method as presented in Section 4.3 is adapted to the charge-oriented Modified Nodal Analysis. Section 4.4 examines the possibility to modify a given circuit in such a way that the DAE that results from either the classical Modified Nodal Analysis or the charge-oriented Modified Nodal Analysis has differentiation index 1. This approach is based on joint work with Falk Ebert.

In the first part of Chapter 5, the academic circuit simulator qPsim is presented. This circuit simulator is based on the charge-oriented Modified Nodal Analysis and includes the index reduction for the resulting DAEs as proposed in Chapter 4. In the second part of this chapter, several examples are considered and the impact of the index reduction on the performance of the DAE solvers implemented in qPsim is studied.

Finally, the results are summarized and some open questions are pointed out.

2. Differential-algebraic equations

Differential-algebraic equations or DAEs in short arise in many application areas such as the simulation of multibody systems, chemical reactions or the transient behavior of circuits. All these applications share the property that a mathematical model consists of both differential equations, that describe the dynamics of the system under consideration, as well as algebraic equations, which model constraints such as mass or energy conservation.

Although there are comprehensive theories for the analytical and numerical treatment of ordinary differential equations (ODE) as well as general nonlinear algebraic equations, the situation is much more complex in the case of DAEs. Whereas linear DAEs with constant coefficients are well understood, even the case of linear DAEs with time-variant coefficients poses some difficulties. For general nonlinear DAEs without special structural properties, many questions are still unanswered.

This chapter gives an introduction to the theory of general nonlinear DAEs. We start by giving the basic definitions in Section 2.1. Section 2.2 will introduce various index concepts which are used to classify DAEs and describe their analytical and numerical properties. Section 2.3 presents two widely used families of discretization methods which are suitable for the numerical integration of DAEs. Section 2.4 presents several methods to reduce the index of DAEs in order to improve their numerical properties. Section 2.5 gives an overview over existing software for the numerical treatment of DAEs.

2.1. Basic definitions

We will start by presenting the basic terminology. The following definitions are taken from [50].

Definition 2.1. (Differential-algebraic equation, DAE) A *general nonlinear differential-algebraic equation (DAE)* is given by an equation

$$\mathbf{f}(t, \mathbf{x}, \dot{\mathbf{x}}) = 0 \tag{2.1a}$$

with $\mathbf{f} : \mathbb{I} \times \mathbb{D}_{\mathbf{x}} \times \mathbb{D}_{\dot{\mathbf{x}}} \to \mathbb{R}^m$. Here, $\mathbb{I} \subseteq \mathbb{R}$ is a compact interval and the domains $\mathbb{D}_{\mathbf{x}}$ and $\mathbb{D}_{\dot{\mathbf{x}}}$ are open subsets of $\mathbb{R}^n$.

If, in addition, the initial values

$$\mathbf{x}(t_0) = \mathbf{x}_0 \in \mathbb{R}^n \tag{2.1b}$$

are given, we call (2.1) an initial value problem. □

If $\frac{\partial}{\partial \dot{\mathbf{x}}}\mathbf{f}$ is nonsingular, then (2.1a) is an implicit ODE and the existing theory and numerical methods for ODEs can be applied. This includes the well-known theorem of Picard and Lindelöf [39] which guarantees the existence and uniqueness of the solution of (2.1) on an interval $\mathbb{I}$ if $\mathbf{f}$ is Lipschitz continuous with respect to $\dot{\mathbf{x}}$ and $\mathbf{x}$ on $\mathbb{I}$.

If on the other hand $\frac{\partial}{\partial \dot{\mathbf{x}}}\mathbf{f} = 0$, then we are faced with a nonlinear algebraic equation in t and $\mathbf{x}$. In this case, a solution of (2.1a) may not exist. Moreover, if a solution exists, then it may not be unique regardless of the initial values (2.1b) (cf. [67]). In addition, we are not free to choose the initial values arbitrarily, since they are already determined by (2.1a).

Hence, if $\frac{\partial}{\partial \dot{\mathbf{x}}}\mathbf{f} \neq 0$ is singular, we may have to deal with some problems regarding the choice of the initial values (2.1b) and the solvability of the initial value problem (2.1), since the problem (2.1) is a mixed system of both differential and algebraic equations.

Example 2.2. Consider the DAE

$$\begin{aligned} 0 &= \dot{x}_1 + x_1 - x_2, \\ 0 &= x_1 + x_2 + x_3, \\ 0 &= x_2 + \cos(t), \end{aligned} \tag{2.2}$$

which consists of one differential and two algebraic equations. Obviously, a solution of (2.2) will only exist if the initial values satisfy both

$$x_{1,0} + x_{3,0} = \cos(t_0) \quad \text{and} \quad x_{2,0} = -\cos(t_0),$$

which leaves one degree of freedom in the choice of initial values. If we choose for example

$$x_{1,0} = 0, \qquad x_{2,0} = -\cos(t_0), \qquad x_{3,0} = \cos(t_0)$$

as initial values, then we obtain the unique solution

$$\begin{aligned} x_1 &= \frac{1}{2}\left[\cos(t) + \sin(t) - e^{-t}\right], \\ x_2 &= -\cos(t), \\ x_3 &= \frac{1}{2}\left[\cos(t) - \sin(t) + e^{-t}\right]. \end{aligned}$$

□

Example 2.3. Consider the DAE

$$\begin{aligned} 0 &= \dot{x}_1 + x_1 - x_2, \\ 0 &= x_1 - \cos(t). \end{aligned} \tag{2.3}$$

Initial values have to fulfill the equation $x_{1,0} = \cos(t_0)$. However, a closer look at the DAE reveals that we cannot choose an arbitrary initial value for x_2 since we have

$$x_{2,0} = \dot{x}_{1,0} + x_{1,0} = -\sin(t_0) + \cos(t_0).$$

Hence, despite the fact that (2.3) includes one differential equation, no initial values can be chosen. □

Example 2.3 shows that the properties of DAEs may differ greatly from the properties of ODEs. These differences motivate the following definitions.

Definition 2.4. (Solution, solvability, consistency) Let $\mathcal{C}^k(\mathbb{D}, \mathbb{R}^m)$ be the vector space of k-times continuously differentiable functions from a domain $\mathbb{D} \subset \mathbb{R}^n$ into $\mathbb{R}^m$.

1. A function $\mathbf{x} \in \mathcal{C}^1(\mathbb{I}, \mathbb{R}^n), \mathbb{I} \subseteq \mathbb{R}$ is called *a solution of the DAE* (2.1a), if it satisfies (2.1a) pointwise.
2. A function $\mathbf{x} \in \mathcal{C}^1(\mathbb{I}, \mathbb{R}^n)$ is *a solution of the initial value problem* (2.1), if $\mathbf{x}$ is a solution of (2.1a) and in addition satisfies (2.1b).
3. The initial values (2.1b) are called *consistent initial values*, if the initial value problem (2.1) has at least one solution.
4. The initial value problem (2.1) is called *solvable*, if it has at least one solution.

□

In the following, we will only consider the case of $m = n$ and assume that (2.1) has a unique solution. We will call such DAEs *regular.*

Definition 2.5. (Quasi-linear DAE) A *quasi-linear DAE* has the form

$$\mathbf{E}(t, \mathbf{x})\dot{\mathbf{x}} + \mathbf{g}(t, \mathbf{x}) = \mathbf{0}, \tag{2.4}$$

where $\mathbf{E}(t, \mathbf{x}) \in \mathcal{C}(\mathbb{I} \times \mathbb{D}_{\mathbf{x}}, \mathbb{R}^{n \times n})$ and $\mathbf{g}(t, \mathbf{x}) \in \mathcal{C}(\mathbb{I} \times \mathbb{D}_{\mathbf{x}}, \mathbb{R}^n)$. □

Quasi-linear DAEs arise in many applications in engineering science. For example, [72] treats DAEs that arise from mechanical multibody systems. As we will see in Chapter 3, the DAEs that arise in circuit simulation also can be written as quasi-linear DAEs. Sometimes it is possible to rewrite (2.4) as

$$\begin{bmatrix} \mathbf{E}_1(t, \mathbf{x}_1, \mathbf{x}_2) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{x}}_1 \\ \dot{\mathbf{x}}_2 \end{bmatrix} + \begin{bmatrix} \mathbf{g}_1(t, \mathbf{x}_1, \mathbf{x}_2) \\ \mathbf{g}_2(t, \mathbf{x}_1, \mathbf{x}_2) \end{bmatrix} = \mathbf{0}, \tag{2.5}$$

where $\mathbf{E}_1(t, \mathbf{x}) \in \mathcal{C}(\mathbb{I} \times \mathbb{D}_{\mathbf{x}_1} \times \mathbb{D}_{\mathbf{x}_2}, \mathbb{R}^{m_1 \times m_1})$ is pointwise nonsingular, $\mathbf{g}_i(t, \mathbf{x}_1, \mathbf{x}_2) \in \mathcal{C}(\mathbb{I} \times \mathbb{D}_{\mathbf{x}_1} \times \mathbb{D}_{\mathbf{x}_2}, \mathbb{R}^{m_i})$ and $\mathbf{x}_i \in \mathbb{R}^{m_i}$, $i = 1, 2$ with $m_1 + m_2 = n$. DAEs of the form (2.5) are called *semi-implicit DAEs.* Semi-implicit DAEs form an important subclass of quasi-linear DAEs and are studied widely because of the additional structure present in (2.5).

Another way to describe DAEs is the concept of DAEs with properly stated leading term [60–62]. A nonlinear DAE (2.1a) has a *properly stated leading term* if it can be written as

$$\begin{aligned} \mathbf{f}(t, \mathbf{x}, \dot{\mathbf{x}}) = \mathbf{A}(t, \mathbf{x})\frac{d}{dt}\mathbf{d}(t, \mathbf{x}) + \mathbf{b}(t, \mathbf{x}) = \mathbf{0}, \\ \mathbf{A} \in \mathcal{C}(\mathbb{I} \times \mathbb{D}_{\mathbf{x}}, \mathbb{R}^{n \times m}), \\ \mathbf{D} := \frac{\partial}{\partial \mathbf{x}}\mathbf{d} \in \mathcal{C}^1(\mathbb{I} \times \mathbb{D}_{\mathbf{x}}, \mathbb{R}^{m \times n}), \end{aligned} \tag{2.6}$$

with $\ker \mathbf{A}(t,\mathbf{x}) \oplus \operatorname{range} \mathbf{D}(t,\mathbf{x}) = \mathbb{R}^m$ and there exists a projector $\mathbf{R} \in \mathcal{C}^1(\mathbb{I}, \mathbb{R}^{m\times m})$ such that

$$\begin{aligned} \ker \mathbf{R}(t) &= \ker \mathbf{A}(t,\mathbf{x}), \\ \operatorname{range} \mathbf{R}(t) &= \operatorname{range} \mathbf{D}(t,\mathbf{x}) \end{aligned}$$

for all $t \in \mathbb{I}$. This formulation has the advantage that a solution $\mathbf{x}$ of (2.6) needs less smoothness than a solution of (2.1a). Due to the splitting of the leading term, the parts of $\mathbf{x}$ that need to be differentiable are clearly visible. This in turn leads to a different definition of the solution of a DAE with properly stated leading term.

Definition 2.6. (Solution of a DAE with properly stated leading term)(cf. [53]) A function $\mathbf{x}$ is called a *solution of* (2.6), if

$$\mathbf{x} \in \mathcal{C}^1_D(\mathbb{I}, \mathbb{R}^n) := \{\mathbf{x} \in \mathcal{C}(\mathbb{I}, \mathbb{R}^n) : \mathbf{D}(t,\mathbf{x})\mathbf{x} \in \mathcal{C}^1(\mathbb{I}, \mathbb{R}^m)\} \tag{2.7}$$

and satisfies (2.6) pointwise. □

2.2. Index concepts

As we have already seen in Example 2.3, the mix of differential and algebraic equations in a DAE may lead to some problems concerning consistent initialization of a given DAE. Moreover, if we take a closer look at (2.3), we see that we have used the derivative of $x_1 = \cos(t)$ to determine a consistent value for x_2. This shows that some variables of a DAE may require higher smoothness than other parts. In addition, the same derivative is necessary to compute a solution of (2.3). Since these derivatives usually are not explicitly part of the DAE, they are computed implicitly during the numerical integration of the DAE as Example 2.7 shows.

Example 2.7. If we apply the backward Euler method to discretize (2.3), we obtain

$$\begin{aligned} 0 &= \frac{x_{1,n+1} - x_{1,n}}{h} + x_{1,n+1} - x_{2,n+1}, \\ 0 &= x_{1,n+1} - \cos(t_{n+1}). \end{aligned}$$

Solving this for $\mathbf{x}_{n+1}$, we have

$$x_{1,n+1} = \cos(t_{n+1}) \quad \text{and} \quad x_{2,n+1} = \frac{\cos(t_{n+1}) - \cos(t_n)}{h} + \cos(t_{n+1}).$$

From this we see that the derivative of x_1 that we need in order to compute the solution for x_2 is approximated by a finite difference that uses the same step size h as the backward Euler method. □

These differences in the analytical and numerical properties of DAEs and ODEs led to the development of various index concepts. These concepts can be seen as methods to measure the distance of a DAE to an ODE with a solution set that includes the solution set of the DAE. In the following, several index concepts that are of importance in the discussion of DAEs from circuit simulation will be introduced.

2.2.1. Differentiation index

One of the first index concepts that has been developed for general nonlinear DAEs is the differentiation index [20, 30, 31]. Its definition has been motivated by the attempt to transform a given DAE with consistent initial value $\mathbf{x}_0$ into an ODE which has the same analytical solution for the initial value $\mathbf{x}_0$ and to apply the existing theory and numerical methods to this ODE. In order to perform this transformation the derivatives of (2.1a) are necessary.

Definition 2.8. (Derivative array)(cf. [18, 20, 21, 30, 31]) Let $\mathbf{f} : \mathbb{I} \times \mathbb{D}_{\mathbf{x}} \times \mathbb{D}_{\dot{\mathbf{x}}} \to \mathbb{R}^n$, $\mathbb{I} \subseteq \mathbb{R}$, $\mathbb{D}_{\mathbf{x}}, \mathbb{D}_{\dot{\mathbf{x}}} \subseteq \mathbb{R}^n$ be a function in $\mathcal{C}^s\left(\mathbb{I} \times \mathbb{D}_{\mathbf{x}} \times \mathbb{D}_{\dot{\mathbf{x}}}, \mathbb{R}^n\right)$. Then, the *derivative array of order $l \leq s$ of* $\mathbf{f}$ is given by

$$\mathbf{F}_l\left(t, \mathbf{x}, \dot{\mathbf{x}}, \ldots, \mathbf{x}^{(l+1)}\right) := \begin{bmatrix} \mathbf{f}(t, \mathbf{x}, \dot{\mathbf{x}}) \\ \frac{d}{dt}\mathbf{f}(t, \mathbf{x}, \dot{\mathbf{x}}) \\ \vdots \\ \frac{d^l}{dt^l}\mathbf{f}(t, \mathbf{x}, \dot{\mathbf{x}}) \end{bmatrix}. \tag{2.8}$$

□

Definition 2.9. (Differentiation index) [15] Let (2.1a) be a regular DAE and assume that $\mathbf{f}$ is sufficiently smooth. The *differentiation index* ν_d is the smallest integer l such that

$$\mathbf{F}_l\left(t, \mathbf{x}, \dot{\mathbf{x}}, \ldots, \mathbf{x}^{(l+1)}\right) = 0 \tag{2.9}$$

uniquely determines $\dot{\mathbf{x}}$ as a function of $\mathbf{x}$ and t. The resulting ODE is called underlying ODE of (2.1a). □

Definition 2.9 basically means that we try to transform the DAE (2.1a) into an ODE

$$\dot{\mathbf{x}} = \bar{\mathbf{f}}_1(t, \mathbf{x}) \tag{2.10a}$$

which has a solution set that includes the analytical solution of (2.1a). During this transformation we may encounter algebraic constraints

$$0 = \bar{\mathbf{f}}_2(t, \mathbf{x}) \tag{2.10b}$$

which have not been explicitly present in the original DAE. These constraints that are only revealed by differentiations and algebraic transformations, are called *hidden constraints*. DAEs with differentiation index $\nu_d \geq 2$ are often referred to as *higher index DAEs*.

2.2.2. Tractability index

The tractability index originally has been developed for linear DAEs with time-variant coefficients [32] and was later generalized to nonlinear DAEs [58]. However, recently the definition has been adapted to DAEs with properly stated leading terms (2.6) [59,61–63]. Here, we will present this newer version of the tractability index as proposed in [63].

Let $\mathbb{D}_{\mathbf{y}} \times \bar{\mathbb{D}}_{\mathbf{x}} \subseteq \mathbb{R}^m \times \mathbb{R}^n$ be an open subset with $\bar{\mathbb{D}}_{\mathbf{x}} \subseteq \mathbb{D}_{\mathbf{x}}$ and $\mathbb{D}_{\mathbf{y}} \subseteq \mathbb{R}^m$ such that $\mathbb{D}_{\mathbf{y}} \cap \operatorname{range} \mathbf{D}(t,\mathbf{x}) \neq \emptyset$ for $t \in \mathbb{I}$ and let $\mathbf{D}^-(t,\mathbf{x}) \in \mathbb{R}^{n\times m}$ be a generalized inverse of $\mathbf{D}(t,\mathbf{x})$ that satisfies

$$\begin{aligned}\mathbf{D}^-(t,\mathbf{x})\mathbf{D}(t,\mathbf{x})\mathbf{D}^-(t,\mathbf{x}) &= \mathbf{D}^-(t,\mathbf{x}),\\ \mathbf{D}(t,\mathbf{x})\mathbf{D}^-(t,\mathbf{x})\mathbf{D}(t,\mathbf{x}) &= \mathbf{D}(t,\mathbf{x}),\\ \mathbf{D}(t,\mathbf{x})\mathbf{D}^-(t,\mathbf{x}) &= \mathbf{R}(t)\end{aligned}$$

for all $t \in \mathbb{I}$. Furthermore, we define the matrices

$$\begin{aligned}\mathbf{G}_0(t,\mathbf{x}) &:= \mathbf{A}(t,\mathbf{x})\mathbf{D}(t,\mathbf{x}),\\ \mathbf{B}_0(t,\mathbf{y},\mathbf{x}) &:= \frac{\partial}{\partial \mathbf{x}}\mathbf{b}(t,\mathbf{x}) + \frac{\partial}{\partial \mathbf{x}}\left[\mathbf{A}(t,\mathbf{x})\mathbf{y}\right].\end{aligned} \tag{2.11a}$$

These matrices are the starting point for a matrix chain that is used to characterize the tractability index of a DAE (2.6). To be able to define the matrices in the next step of this chain, we introduce two projector functions

$$\mathbf{P}_0(t) := \mathbf{D}^-(t,\mathbf{x})\mathbf{D}(t,\mathbf{x}) \qquad \text{and} \qquad \mathbf{Q}_0(t) = \mathbf{I} - \mathbf{P}_0(t).$$

From the definitions of $\mathbf{D}^-(t,\mathbf{x})$ and $\mathbf{G}_0(t,\mathbf{x})$ we see that $\mathbf{Q}_0(t)$ is a projector onto $\ker \mathbf{G}_0(t,\mathbf{x})$. With these projector functions we are able to define

$$\mathbf{G}_1(t,\mathbf{y},\mathbf{x}) = \mathbf{G}_0(t,\mathbf{x}) + \mathbf{B}_0(t,\mathbf{y},\mathbf{x})\mathbf{Q}_0(t). \tag{2.11b}$$

Now, we choose $\mathbf{Q}_1(t,\mathbf{y},\mathbf{x})$ to be a projector onto $\ker \mathbf{G}_1(t,\mathbf{y},\mathbf{x})$ and set $\mathbf{P}_1(t,\mathbf{y},\mathbf{x}) = \mathbf{I} - \mathbf{Q}_1(t,\mathbf{y},\mathbf{x})$, $\mathbf{y} \in \mathbb{R}^m$, $\mathbf{x} \in \bar{\mathbb{D}}_{\mathbf{x}}$, $t \in \mathbb{I}$. Note that these projectors both depend on $\mathbf{y}$ and $\mathbf{x}$. However, we will assume that the product

$$\mathbf{D}(t,\mathbf{x})\mathbf{P}_0(t)\mathbf{P}_1(t,\mathbf{y},\mathbf{x})\mathbf{D}^-(t,\mathbf{x}) = \mathbf{D}(t,\mathbf{x})\mathbf{P}_1(t,\mathbf{y},\mathbf{x})\mathbf{D}^-(t,\mathbf{x})$$

is independent of $\mathbf{y}$. We will drop this argument in the following and write $(\mathbf{D}\mathbf{P}_1\mathbf{D}^-)(t,\mathbf{x})$. With this we are able to introduce

$$\mathbf{B}_1(t,\dot{\mathbf{x}},\mathbf{y},\mathbf{x}) := \mathbf{B}_0(t,\mathbf{y},\mathbf{x})\mathbf{P}_0(t) - \mathbf{G}_1(t,\mathbf{y},\mathbf{x})\mathbf{D}^-(t,\mathbf{x})\frac{d}{dt}\left[\left(\mathbf{D}\mathbf{P}_1\mathbf{D}^-\right)(t,\mathbf{x})\right]\mathbf{D}(t,\mathbf{x}) \tag{2.11c}$$

and

$$\mathbf{G}_2(t,\dot{\mathbf{x}},\mathbf{y},\mathbf{x}) := \mathbf{G}_1(t,\mathbf{y},\mathbf{x}) + \mathbf{B}_1(t,\dot{\mathbf{x}},\mathbf{y},\mathbf{x})\mathbf{Q}_1(t,\mathbf{y},\mathbf{x}). \tag{2.11d}$$

For the subsequent matrix pairs $\mathbf{B}_i(t,\dot{\mathbf{x}},\mathbf{y},\mathbf{x})$ and $\mathbf{G}_{i+1}(t,\dot{\mathbf{x}},\mathbf{y},\mathbf{x})$ we choose $\mathbf{Q}_i(t,\dot{\mathbf{x}},\mathbf{y},\mathbf{x})$ as a projector onto $\ker \mathbf{G}_i(t,\dot{\mathbf{x}},\mathbf{y},\mathbf{x})$ and set $\mathbf{P}_i(t,\dot{\mathbf{x}},\mathbf{y},\mathbf{x}) = \mathbf{I} - \mathbf{Q}_i(t,\dot{\mathbf{x}},\mathbf{y},\mathbf{x})$. With this and in analogy to (2.11c) and (2.11d) we define for $i > 1$

$$\begin{aligned}\mathbf{B}_i(t,\dot{\mathbf{x}},\mathbf{y},\mathbf{x}) := {} & \mathbf{B}_{i-1}(t,\mathbf{y},\mathbf{x})\mathbf{P}_{i-1}(t,\dot{\mathbf{x}},\mathbf{y},\mathbf{x}) \\ & - \mathbf{G}_i(t,\dot{\mathbf{x}},\mathbf{y},\mathbf{x})\mathbf{D}^-(t,\mathbf{x})\frac{d}{dt}\left[\left(\mathbf{D}\left(\prod_{j=1}^{i}\mathbf{P}_j\right)\mathbf{D}^-\right)(t,\mathbf{x})\right]\left(\mathbf{D}\prod_{j=1}^{i-1}\mathbf{P}_j\right)(t,\mathbf{x})\end{aligned} \tag{2.11e}$$

and

$$\mathbf{G}_{i+1}(t,\dot{\mathbf{x}},\mathbf{y},\mathbf{x}) := \mathbf{G}_i(t,\dot{\mathbf{x}},\mathbf{y},\mathbf{x}) + \mathbf{B}_i(t,\dot{\mathbf{x}},\mathbf{y},\mathbf{x})\mathbf{Q}_i(t,\dot{\mathbf{x}},\mathbf{y},\mathbf{x}), \tag{2.11f}$$

pointwise for $\dot{\mathbf{x}} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^m$, $\mathbf{x} \in \bar{\mathbb{D}}_{\mathbf{x}}$, $t \in \mathbb{I}$.

Definition 2.10. (Tractability index) The DAE (2.6) has *tractability index* ν_t if there is a sequence of matrices defined by (2.11a)–(2.11f) such that for $i \geq 0$, $t \in \mathbb{I}$, $\mathbf{x} \in \bar{\mathbb{D}}_{\mathbf{x}}$, $\mathbf{y} \in \mathbb{D}_{\mathbf{y}}$, $\dot{\mathbf{x}} = \mathbf{D}^-(t,\mathbf{x})(\mathbf{y} - \mathbf{D}'(t)\mathbf{x}) + \mathbf{z}$, $\mathbf{z} \in \ker \mathbf{G}_0(t,\mathbf{x})$ it holds that

1. $\operatorname{rank} \mathbf{G}_i(t,\dot{\mathbf{x}},\mathbf{y},\mathbf{x}) = r_i$,
2. $\mathbf{Q}_i(t,\dot{\mathbf{x}},\mathbf{y},\mathbf{x})\mathbf{Q}_j(t,\dot{\mathbf{x}},\mathbf{y},\mathbf{x}) = 0$ for $j = 0,\dots,i-1$,
3. $\mathbf{Q}_i$ is continuous and $\mathbf{D}\left(\prod_{j=1}^{i}\mathbf{P}_j\right)\mathbf{D}^-$, $i \geq 1$ is continously differentiable and does not depend on $\dot{\mathbf{x}}$ and $\mathbf{y}$,
4. $r_0 \leq r_1 \leq \cdots \leq r_{\nu_t - 1} < m$ and $r_{\nu_t} = m$.

□

Remark 2.11. *In contrast to the definition of the differentiation index, the tractability index does not require higher derivatives of the respective DAE. However, the projectors* $\mathbf{P}_i$ *may depend implicitly on derivatives of* $\mathbf{P}_0$ *and may be hard to actually compute. For DAEs with tractability index greater than* 2*, these derivatives may be difficult to determine in practice.*

A numerical algorithm to determine the tractability index of linear DAEs is described in [56]. The same algorithm can be applied to linearized nonlinear DAEs to compute the tractability index.

2.2.3. Strangeness index

The strangeness index is a generalization of the differentiation index that also allows the treatment of over- and underdetermined DAEs [47,49]. Moreover, existence and uniqueness results for the solution of (2.1) require less smoothness of (2.1) in the strangeness index framework than in the differentiation index framework [43,44,46]. At the same time the strangeness index approach can be used to derive numerical methods to treat higher index DAEs [45,52,55].

To define the strangeness index we consider again the derivative array (2.8). In addition to this we define the Jacobians

$$\mathbf{M}_l = \mathbf{M}_l(t, \mathbf{x}, \dot{\mathbf{x}}, \dots, \mathbf{x}^{(l+1)}) = \mathbf{F}_{l;\dot{\mathbf{x}},\dots,\mathbf{x}^{(l+1)}}(t, \mathbf{x}, \dot{\mathbf{x}}, \dots, \mathbf{x}^{(l+1)}) \tag{2.12a}$$

and

$$\mathbf{N}_l = \mathbf{N}_l(t, \mathbf{x}, \dot{\mathbf{x}}, \dots, \mathbf{x}^{(l+1)}) = -\left[\mathbf{F}_{l;\mathbf{x}}(t, \mathbf{x}, \dot{\mathbf{x}}, \dots, \mathbf{x}^{(l+1)})\ \mathbf{0} \cdots \mathbf{0}\right] \tag{2.12b}$$

of size $ln \times ln$.

The strangeness index concept relies on some constant rank assumptions that are summed up in Hypothesis 2.12. The solution set

$$\mathbb{L}_l := \{(t, \mathbf{x}, \dots, \mathbf{x}^{(l+1)}) \in \mathbb{R}^{(l+2)n+1} : \mathbf{F}_l(t, \mathbf{x}, \dots, \mathbf{x}^{(l+1)}) = 0\}$$

which will be used in the hypothesis is associated with the derivative array $\mathbf{F}_l$ of $\mathbf{f}$ of order l.

Hypothesis 2.12. *(cf. [46, 50]) Consider a regular DAE* (2.1a). *There exist integers ν, a, and d such that $\mathbb{L}_\nu$ is nonempty and that for every $(t_0, \mathbf{x}_0, \dots, \mathbf{x}_0^{(\nu+1)}) \in \mathbb{L}_\nu$ there exists a neighborhood in which the following conditions are satisfied.*

1. *The Jacobian $\mathbf{M}_\nu$ has constant rank $(\nu+1)n - a$ on $\mathbb{L}_\nu$ such that there exists a smooth matrix function $\mathbf{Z}_2 : \mathbb{R}^{(\nu+2)n+1} \longmapsto \mathbb{R}^{(\nu+1)n \times a}$ which has pointwise maximal rank and which satisfies $\mathbf{Z}_2^T \mathbf{M}_\nu = \mathbf{0}$ on $\mathbb{L}_\nu$.*
2. *The matrix function $\widehat{\mathbf{A}}_2(t, \mathbf{x}, \dots, \mathbf{x}^{(\nu+1)}) = \mathbf{Z}_2^T \mathbf{N}_\nu \left[\mathbf{I}_n\ \mathbf{0} \dots \mathbf{0}\right]^T$ has constant rank a on $\mathbb{L}_\nu$ such that there exists a smooth matrix function $\mathbf{T}_2 : \mathbb{R}^{(\nu+2)n+1} \longmapsto \mathbb{R}^{n \times d}$, $d = n - a$ which has pointwise maximal rank and which satisfies $\widehat{\mathbf{A}}_2 \mathbf{T}_2 = \mathbf{0}$.*
3. *The matrix function $\mathbf{f}_{\dot{\mathbf{x}}}(t, \mathbf{x}, \dot{\mathbf{x}}) \mathbf{T}_2(t, \mathbf{x}, \dots, \mathbf{x}^{(\nu+1)})$ has constant rank d on $\mathbb{L}_\nu$ such that there exists a smooth matrix function $\mathbf{Z}_1 : \mathbb{R}^{(\nu+2)n+1} \longmapsto \mathbb{R}^{n \times d}$ which has pointwise maximal rank and which satisfies* $\operatorname{rank} \widehat{\mathbf{E}}_1 \mathbf{T}_2 = d$*, where $\widehat{\mathbf{E}}_1 = \mathbf{Z}_1^T \mathbf{f}_{\dot{\mathbf{x}}}$.*

Remark 2.13. *It is possible to generalize Hypothesis 2.12 to more general DAEs (cf. [47, 50]).*

Definition 2.14. (Strangeness index, strangeness-free) Consider a regular nonlinear DAE (2.1a). The smallest integer ν_s for which (2.1a) satisfies Hypothesis 2.12 is called the *strangeness index of* (2.1a). If $\nu_s = 0$, then (2.1a) is called *strangeness-free.* □

2.3. Numerical integration methods for DAEs

In this section, we consider the numerical integration of semi-explicit DAEs (2.13)

$$0 = \dot{\mathbf{x}} + \mathbf{g}_1(t, \mathbf{x}, \mathbf{y}), \tag{2.13a}$$
$$0 = \mathbf{g}_2(t, \mathbf{x}, \mathbf{y}), \tag{2.13b}$$

with consistent initial values $(\mathbf{x}(t_0), \mathbf{y}(t_0)) = (\mathbf{x}_0, \mathbf{y}_0)$ over the integration interval $\mathbb{I} = [t_0, t_N] \subset \mathbb{R}$. For the most part we will restrict the discussion to DAEs with differentiation index 1.

Considering the numerical integration of initial value problems in the case of ODEs

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(t, \mathbf{x}), \\ \mathbf{x}(t_0) &= \mathbf{x}_0, \end{aligned} \tag{2.14}$$

existing methods include one-step methods and multi-step methods which we will discuss in this section in more detail.

We consider an equidistant discretization of the integration interval $\mathbb{I}$ with stepsize $h = \frac{t_N - t_0}{N}$ and write $t_i = t_0 + i \cdot h$. The exact solution $\mathbf{x}(t_i)$ at t_i is denoted by $\mathbf{x}_i$ and the numerical approximation to $\mathbf{x}_i$ by $\boldsymbol{\xi}_i$. A one-step method tries to advance the numerical solution from $\boldsymbol{\xi}_i$ to $\boldsymbol{\xi}_{i+1}$ by only using the previous step $\boldsymbol{\xi}_i$. Generally, these kinds of methods can be written as

$$\boldsymbol{\xi}_{i+1} = \boldsymbol{\xi}_i + h\boldsymbol{\phi}(t_i, \boldsymbol{\xi}_i; h; \mathbf{f}), \; i = 0, \ldots, N-1 \tag{2.15}$$

with an increment function $\boldsymbol{\phi} = \boldsymbol{\phi}(t, \mathbf{x}; h; \mathbf{f}) = \boldsymbol{\phi}(t, \mathbf{x}; h)$ [73]. An important class of one-step methods is given by Runge-Kutta methods. The properties of these methods and their application to DAE initial value problems will be discussed in Subsection 2.3.1.

Multi-step methods on the other hand not only use the approximated solution $\boldsymbol{\xi}_i$ of the last step to compute $\boldsymbol{\xi}_{i+1}$, but include past values $\boldsymbol{\xi}_i, \boldsymbol{\xi}_{i-1}, \ldots, \boldsymbol{\xi}_{i-r+1}$. In Subsection 2.3.2 we will discuss BDF methods which are linear multi-step methods of the form

$$\sum_{j=0}^{r} \alpha_{r-j} \boldsymbol{\xi}_{i+1-j} = h \sum_{j=0}^{r} \beta_{r-j} \mathbf{f}(t_{i+1-j}, \boldsymbol{\xi}_{i+1-j}), \tag{2.16}$$

with coefficients α_j, β_j and $\alpha_r \neq 0$.

Before we start the discussion, we give some basic definitions. To do so, we note that for a fixed stepsize h both one-step and multi-step methods generate sequences of approximations to the solution of (2.14) that satisfy an iteration

$$\mathbf{X}_{i+1} = \boldsymbol{\Psi}(t_i, \mathbf{X}_i; h) \tag{2.17}$$

with $\mathbf{X}_i \in \mathbb{R}^{\mathcal{N}}$ (cf. [50]). The actual solution at t_i is given by $\mathbf{X}(t_i) \in \mathbb{R}^{\mathcal{N}}$. For general one-step methods, we are able to set $\mathbf{X}_i = \boldsymbol{\xi}_i$ and $\boldsymbol{\Psi}(t_i, \mathbf{X}_i; h) = \boldsymbol{\xi}_i + h\boldsymbol{\phi}(t_i, \boldsymbol{\xi}_i; h)$ to write equation (2.15) in the form (2.17). If we consider linear multistep methods (2.16), then we set

$$\mathbf{X}_i = \begin{bmatrix} \boldsymbol{\xi}_i \\ \boldsymbol{\xi}_{i-1} \\ \vdots \\ \boldsymbol{\xi}_{i-r+1} \end{bmatrix} \text{ and } \mathbf{X}(t_i) = \begin{bmatrix} \mathbf{x}_i \\ \mathbf{x}_{i-1} \\ \vdots \\ \mathbf{x}_{i-r+1} \end{bmatrix}.$$

Moreover, we are able to solve (2.16) for $\boldsymbol{\xi}_{i+1}$ by the Implicit Function Theorem. With this we obtain

$$\boldsymbol{\xi}_{i+1} = \mathcal{S}(t_{i+1}, \boldsymbol{\xi}_i, \boldsymbol{\xi}_{i-1}, \ldots, \boldsymbol{\xi}_{i-r+1}; h),$$

which allows us to define

$$\boldsymbol{\Psi}(t_i, \mathbf{X}_i; h) := \begin{bmatrix} \mathcal{S}(t_{i+1}, \boldsymbol{\xi}_i, \boldsymbol{\xi}_{i-1}, \ldots, \boldsymbol{\xi}_{i-r+1}; h) \\ \boldsymbol{\xi}_i \\ \boldsymbol{\xi}_{i-1} \\ \vdots \\ \boldsymbol{\xi}_{i-r+2} \end{bmatrix}$$

and to rewrite equation (2.16) in terms of (2.17).

Therefore it is possible to consider *general discretization methods* of the form (2.17) in the following. The *local discretization error* of a general discretization method

$$\tau(\mathbf{x}, t_{i+1}; h) := \|\mathbf{X}(t_{i+1}) - \boldsymbol{\Psi}(t_i, \mathbf{X}(t_i); h)\| \tag{2.18}$$

is a measure for how well the exact solution $\mathbf{x}(t)$ of (2.14) satisfies (2.17).

Definition 2.15. (Consistency of order p) [50] A general discretization method (2.17) is *consistent of order* p if there exists a constant C such that

$$\tau(\mathbf{x}, t_{i+1}; h) \leq Ch^{p+1}.$$

The constant C does not depend on the stepsize h. □

The *global discretization error at* t_i of (2.17) is given by

$$\epsilon(\mathbf{x}, t_i; h) := \|\mathbf{X}_i - \mathbf{X}(t_i)\|. \tag{2.19}$$

Definition 2.16. (Convergence of order p) [50] A general discretization method (2.17) is *convergent of order* p if there exists a constant C such that

$$\epsilon(\mathbf{x}, t_N; h) \leq Ch^p,$$

provided that the initial error $\epsilon(\mathbf{x}, t_0; h)$ satisfies

$$\epsilon(\mathbf{x}, t_0; h) \leq \widetilde{C}h^p$$

with a constant $\widetilde{C}$ independent of h. □

Definition 2.17. (Stability) [50] A general discretization method is *stable* if there exists a constant K such that for some vector norm $\|\cdot\|$

$$\|\boldsymbol{\Psi}(t_i, \mathbf{X}(t_i); h) - \boldsymbol{\Psi}(t_i, \mathbf{X}_i; h)\| \leq (1 + hK)\|\mathbf{X}(t_i) - \mathbf{X}_i\|$$

with K being independent of the stepsize h. □

Stability will be discussed in greater detail for Runge-Kutta methods later in this section.

Theorem 2.18. *Consider a general discretization method* (2.17). *If the method is stable and consistent of order p, then the method is convergent of order p when applied to an ODE* (2.14).

Proof. A proof of this theorem can be found in [50]. □

2.3.1. Runge-Kutta methods

Discretization of DAEs by Runge-Kutta methods

The discretization of an ODE (2.14) with an s-stage Runge-Kutta method takes the form

$$\boldsymbol{\xi}_{n+1} = \boldsymbol{\xi}_n + h \sum_{i=1}^{s} \beta_j \Xi'_{ni}, \; n = 0, \dots, N-1, \tag{2.20a}$$

where the *internal stages* Ξ_{ni} are given by

$$\Xi_{ni} = \boldsymbol{\xi}_n + h \sum_{j=1}^{s} \alpha_{ij} \Xi'_{nj}, \; i = 1, \dots, s, \tag{2.20b}$$

with *stage derivatives* Ξ'_{ni} defined as

$$\Xi'_{ni} = \mathbf{f}(t_n + \gamma_i h, \Xi_{ni}), \; i = 1, \dots, s. \tag{2.20c}$$

If we define $\boldsymbol{\beta} := [\beta_1, \dots, \beta_s]^T$, $\boldsymbol{\gamma} := [\gamma_1, \dots, \gamma_s]^T$ and $\boldsymbol{\mathcal{A}} = (\alpha_{i,j})_{i,j=1,\dots,s}$, then the method can be written in a compact way as a *Butcher tableau* (2.21).

$$\begin{array}{c|c} \boldsymbol{\gamma} & \boldsymbol{\mathcal{A}} \\ \hline & \boldsymbol{\beta}^T \end{array} \tag{2.21}$$

It is possible to construct various Runge-Kutta methods by choosing $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$. As a first restriction to the choice of coefficients we compare the solution of ODE (2.14) with the solution of an equivalent autonomous problem

$$\dot{\widetilde{\mathbf{x}}} = \widetilde{\mathbf{f}}(\widetilde{\mathbf{x}}), \tag{2.22a}$$

$$\widetilde{\mathbf{x}}(t_0) = \widetilde{\mathbf{x}}_0. \tag{2.22b}$$

To obtain (2.22) from the ODE (2.14), we need to add the trivial equation $\dot{t} = 1$ and the initial value $t(t_0) = t_0$ to (2.14). This yields

$$\dot{\widetilde{\mathbf{x}}} =: \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{t} \end{bmatrix} = \begin{bmatrix} \mathbf{f}(t, \mathbf{x}) \\ 1 \end{bmatrix} := \widetilde{\mathbf{f}}(\widetilde{\mathbf{x}}), \tag{2.23a}$$

$$\widetilde{\mathbf{x}}(t_0) =: \begin{bmatrix} \mathbf{x}(t_0) \\ t(t_0) \end{bmatrix} = \begin{bmatrix} \mathbf{x}_0 \\ t_0 \end{bmatrix} := \widetilde{\mathbf{x}}_0, \tag{2.23b}$$

which has the same analytical solution for $\mathbf{x}$ as (2.14). A Runge-Kutta method for (2.23) takes the form

$$\begin{bmatrix} \mathbf{x}_{n+1} \\ t_{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_n \\ t_n \end{bmatrix} + h \sum_{i=1}^{s} \beta_j \begin{bmatrix} \boldsymbol{\Xi}'_{ni} \\ T'_{ni} \end{bmatrix}, \; n = 0, \dots, N-1, \tag{2.24a}$$

$$\begin{bmatrix} \boldsymbol{\Xi}_{ni} \\ T_{ni} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\xi}_n \\ t_n \end{bmatrix} + h \sum_{j=1}^{s} \alpha_{ij} \begin{bmatrix} \boldsymbol{\Xi}'_{nj} \\ T'_{nj} \end{bmatrix}, \; i = 1, \dots, s, \tag{2.24b}$$

$$\begin{bmatrix} \boldsymbol{\Xi}'_{ni} \\ T'_{nj} \end{bmatrix} = \begin{bmatrix} \mathbf{f}(T_{ni}, \boldsymbol{\Xi}_{ni}) \\ 1 \end{bmatrix}, \; i = 1, \dots, s. \tag{2.24c}$$

Using (2.24c) and (2.24b), it is possible to compute the stage variables

$$T_{ni} = t_n + h \sum_{j=1}^{s} \alpha_{ij}.$$

Inserting this into the first component of (2.24c) yields

$$\boldsymbol{\Xi}'_{ni} = \mathbf{f}(t_n + h \sum_{j=1}^{s} \alpha_{ij}, \boldsymbol{\Xi}_{ni}),$$

and we see that we will only obtain the same numerical solution for $\mathbf{x}$ in (2.14) and (2.23), if

$$\gamma_i = \sum_{j=1}^{s} \alpha_{ij}, \; i = 1, \dots, s \tag{2.25}$$

holds. Hence, we will assume that the Runge-Kutta methods that we consider in the scope of this section satisfy (2.25). The remaining freedom in the choice of coefficients is used to produce methods with specific consistency, convergence and stability properties. This will be discussed later.

If an implicit equation of the form $\mathbf{f}(t, \mathbf{x}, \dot{\mathbf{x}}) = 0$ is considered, then the stage derivatives cannot be computed by (2.20c). Instead, (2.20b) and (2.20c) are replaced by the nonlinear system

$$0 = \mathbf{f}(t_n + \gamma_i h, \boldsymbol{\xi}_n + h \sum_{j=1}^{s} \alpha_{ij} \Xi'_{nj}, \Xi'_{ni}),\ i = 1, \dots, s. \tag{2.26}$$

Obviously, the stage derivatives can be computed in the same way in the case that $\frac{\partial}{\partial \mathbf{x}}\mathbf{f}$ is singular, such that (2.20a) together with (2.26) yields a way to apply Runge-Kutta methods to DAEs. However, in the DAE case the solvability of (2.26) has to be ensured. It has been shown in [35,38,50] that for regular, strangeness-free DAEs where $\mathbf{f}$ is sufficiently smooth, the nonlinear system (2.26) is uniquely solvable for sufficiently small stepsizes h, if $\boldsymbol{\mathcal{A}}$ is nonsingular, i.e. the applied method is implicit.

Stability of Runge-Kutta methods

In the discussion of numerical integration methods for DAEs, stability is an important issue. Roughly speaking stability of a system means that small errors in the input data to the system will result in small errors in the output data. When discussing the stability behavior of one-step methods such as Runge-Kutta methods, Dahlquist's test equation [22]

$$\dot{x} = \lambda x,\ x(t_0) = x_0,\ \lambda \in \mathbb{C},\ \mathbb{Re}(\lambda) < 0. \tag{2.27}$$

plays an important role.

Remark 2.19.

- *In [73] the test equation is given as a system of ordinary differential equations*

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x},\ \mathbf{x}(t_0) = \mathbf{x}_0,$$

where all eigenvalues λ_j, $j = 1, \dots, n$ of $\mathbf{A} \in \mathbb{C}^{n \times n}$ have negative real parts. However, this is just a generalization of (2.27) *to an ODE system.*

- *In [51] the test equation*

$$\begin{bmatrix} 1 & -\omega t \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \lambda & \omega(1 - \lambda t) \\ -1 & 1 + \omega t \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{2.28}$$

for DAEs is defined. Again, we have $\lambda \in \mathbb{C}$, $\mathbb{Re}(\lambda) < 0$. The nullspace of leading term of (2.28) *is rotating with frequency ω. Hence,* (2.28) *allows to study instabilities of Runge-Kutta methods that are caused by such rotating nullspaces.*

Definition 2.20. (Stability function of a discretization method) The *stability function $R(z)$ of a discretization method* (2.17) is given by

$$x_{i+1} = R(z) x_i \tag{2.29}$$

where (2.29) is obtained by applying the method (2.17) to Dahlquist's test equation (2.27) with $z = h\lambda$. □

Proposition 2.21. *The stability function of an implicit s-stage Runge-Kutta method is given by*

$$R(z) = 1 - z\boldsymbol{\beta}^T(\mathbf{I}_s - z\boldsymbol{\mathcal{A}})^{-1}\mathbf{1}$$

with $\mathbf{1} = [1 \dots 1]^T \in \mathbb{R}^s$.

Proof. See [36]. □

Definition 2.22. (A-stability) An implicit Runge-Kutta method whose stability function $R(z)$ satisfies

$$|R(z)| \leq 1 \text{ for all } z \in \mathbb{C} \text{ with } \mathbb{R}\mathrm{e}(z) \leq 0$$

is called *A-stable.* □

Definition 2.23. (L-stability) An implicit Runge-Kutta method that is A-stable and whose stability function in addition satisfies

$$\lim_{\mathbb{R}\mathrm{e}(z) \to -\infty} R(z) = 0.$$

is called *L-stable.* □

Proposition 2.24. *If an implicit Runge-Kutta method with nonsingular $\boldsymbol{\mathcal{A}}$ satisfies one of the conditions*

$$\alpha_{sj} = \beta_j, \;\; j = 1, \dots, s, \tag{2.30a}$$

$$\alpha_{i1} = \beta_1, \;\; i = 1, \dots, s, \tag{2.30b}$$

then $R(\infty) = 0$. This makes A-stable methods L-stable.

Proof. See [36]. □

Methods for with (2.30a) holds are called *stiffly accurate.* Condition (2.30a) means that

$$\begin{aligned} \boldsymbol{\Xi}_{ns} &= \boldsymbol{\xi}_n + h \sum_{j=1}^{s} \alpha_{sj} \boldsymbol{\Xi}'_{nj} \\ &= \boldsymbol{\xi}_n + h \sum_{j=1}^{s} \beta_j \boldsymbol{\Xi}'_{nj} \\ &= \boldsymbol{\xi}_{n+1}, \end{aligned}$$

i.e. the last stage $\boldsymbol{\Xi}_{ns}$ coincides with the numerical approximation of the solution $\boldsymbol{\xi}_{n+1}$. If we consider semi-explicit DAEs (2.13)

$$\begin{aligned} 0 &= \dot{\mathbf{x}} + \mathbf{g}_1(t, \mathbf{x}, \mathbf{y}), \\ 0 &= \mathbf{g}_2(t, \mathbf{x}, \mathbf{y}), \end{aligned}$$

and apply an implicit Runge-Kutta method, then the internal stages $\boldsymbol{\Xi}_{ni}$, $\boldsymbol{\Theta}_{ni}$, $i = 1, \dots, s$ are computed by

$$\begin{aligned}\boldsymbol{\Xi}_{ni} &= \boldsymbol{\xi}_n + h \sum_{j=1}^{s} \alpha_{ij} \mathbf{g}_1(t_{nj}, \boldsymbol{\Xi}_{nj}, \boldsymbol{\Theta}_{nj}),\\ \mathbf{0} &= \mathbf{g}_2(t, \boldsymbol{\Xi}_{nj}, \boldsymbol{\Theta}_{nj}).\end{aligned}$$

Hence, the internal stages are consistent. However, a linear combination of the stages as in (2.20a) is not necessarily consistent. Since for stiffly accurate Runge-Kutta methods the numerical approximation is not a linear combination but equal to the last stage $\boldsymbol{\Xi}_{ns}$, this ensures the consistency of the numerical solution if stiffly accurate Runge-Kutta methods are used for the discretization. [38, 50].

Convergence of Runge-Kutta methods

Up to now the only restriction for the coefficients of an implicit Runge-Kutta method is given by (2.25). Now, we use the remaining freedom to obtain methods of a certain order of convergence. The following order conditions and Theorem 2.25 are due to Butcher [16].

$$B(q): \qquad \sum_{j=1}^{s} \beta_j \gamma_j^{p-1} = \frac{1}{p}, \qquad p = 1, \dots, q \tag{2.31a}$$

$$C(\eta): \qquad \sum_{j=1}^{s} \alpha_{ij} \gamma_j^{q-1} = \frac{\gamma_i^q}{q}, \qquad i = 1, \dots, s,\ q = 1, \dots, \eta \tag{2.31b}$$

$$D(\vartheta): \qquad \sum_{i=1}^{s} \beta_i \gamma_i^{q-1} \alpha_{ij} = \frac{\beta_j}{q}\left(1 - \gamma_j^q\right), \qquad j = 1, \dots, s,\ q = 1, \dots, \vartheta \tag{2.31c}$$

The order conditions (2.31a) and (2.31b) can be interpreted in the following way [35]:

- $B(q)$ means that polynomials up to degree $q-1$ are integrated exactly on the interval $[0, 1]$ by the quadrature formula with weights $\beta_1, \dots, \beta_s$ and nodes $\gamma_1, \dots, \gamma_s$.
- $C(\eta)$ means that for each $i = 1, \dots, s$ polynomials up to degree at least $\eta - 1$ are integrated exactly on the interval $[0, \gamma_i]$ by the quadrature formula with weights $\alpha_{i1}, \dots, \alpha_{is}$.

Theorem 2.25. *If the coefficients of an implicit Runge-Kutta method satisfy* (2.31) *with* $p \leq 2\eta + 2$ *and* $p \leq \vartheta + \eta + 1$*, then the method is consistent of order* p *when applied to ODEs. Hence the method also is convergent of order* p*.*

Proof. See [16, 17] and also [36]. □

$\frac{1}{2}$	$\frac{1}{2}$
	1

$\frac{1}{2}-\frac{\sqrt{3}}{6}$	$\frac{1}{4}$	$\frac{1}{4}-\frac{\sqrt{3}}{6}$
$\frac{1}{2}+\frac{\sqrt{3}}{6}$	$\frac{1}{4}+\frac{\sqrt{3}}{6}$	$\frac{1}{4}$
	$\frac{1}{2}$	$\frac{1}{2}$

$\frac{1}{2}-\frac{\sqrt{15}}{10}$	$\frac{5}{36}$	$\frac{2}{9}-\frac{\sqrt{15}}{15}$	$\frac{5}{36}-\frac{\sqrt{15}}{30}$
$\frac{1}{2}$	$\frac{5}{36}+\frac{\sqrt{15}}{24}$	$\frac{2}{9}$	$\frac{5}{36}-\frac{\sqrt{15}}{24}$
$\frac{1}{2}+\frac{\sqrt{15}}{10}$	$\frac{5}{36}+\frac{\sqrt{15}}{30}$	$\frac{2}{9}+\frac{\sqrt{15}}{15}$	$\frac{5}{36}$
	$\frac{5}{18}$	$\frac{4}{9}$	$\frac{5}{18}$

Table 2.1.: Butcher tableaus for Gauss methods of order 2, 4 and 6

Table 2.1 shows the Butcher tableaus of the 1-, 2- and 3-stage Gauss methods. These methods satisfy $B(2s)$, $C(s)$ and $D(s)$ and are of order $2s$.

The methods shown in Table 2.2 belong to the group of Radau IA methods which satisfy $B(2s-1)$, $C(s-1)$, $D(s)$ and $\gamma_1 = 0$. Moreover, we have (2.30b) and hence the methods are L-stable.

If we require $B(2s-1)$, $C(s)$, $D(s-1)$ and $\gamma_s = 1$ to be satisfied, we obtain methods of the Radau IIA type. By Theorem 2.25, these methods are of order $2s-1$. In addition, the methods are A-stable and stiffly accurate since the coefficients also satisfy (2.30a). Table 2.3 shows the Butcher tableaus of the 1-, 2- and 3-stage Radau IIA methods.

0	1
	1

0	$\frac{1}{4}$	$-\frac{1}{4}$
$\frac{2}{3}$	$\frac{1}{4}$	$\frac{5}{12}$
	$\frac{1}{4}$	$\frac{3}{4}$

0	$\frac{1}{9}$	$\frac{-1-\sqrt{6}}{18}$	$\frac{-1+\sqrt{6}}{18}$
$\frac{6-\sqrt{6}}{10}$	$\frac{1}{9}$	$\frac{88+7\sqrt{6}}{360}$	$\frac{88-43\sqrt{6}}{360}$
$\frac{6+\sqrt{6}}{10}$	$\frac{1}{9}$	$\frac{88+43\sqrt{6}}{360}$	$\frac{88-7\sqrt{6}}{360}$
	$\frac{1}{9}$	$\frac{16+\sqrt{6}}{36}$	$\frac{16-\sqrt{6}}{36}$

Table 2.2.: Butcher tableaus for Radau IA methods of order 1, 3 and 5

1	1
	1

$\frac{1}{3}$	$\frac{5}{12}$	$-\frac{1}{12}$
1	$\frac{3}{4}$	$\frac{1}{4}$
	$\frac{3}{4}$	$\frac{1}{4}$

$\frac{4-\sqrt{6}}{10}$	$\frac{88-7\sqrt{6}}{360}$	$\frac{296-169\sqrt{6}}{1800}$	$\frac{-2+3\sqrt{6}}{225}$
$\frac{4+\sqrt{6}}{10}$	$\frac{296+169\sqrt{6}}{1800}$	$\frac{88+7\sqrt{6}}{360}$	$\frac{-2-3\sqrt{6}}{225}$
1	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$
	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$

Table 2.3.: Butcher tableaus for Radau IIA methods of order 1, 3 and 5

The order of convergence for ODEs is also called *classical order* of the method. If we apply Runge-Kutta methods to DAEs, then the convergence behavior may change drastically. Here, we only consider DAEs of the from (2.13) with differentiation index $\nu_d = 1$. For these DAEs the order of convergence of implicit Runge-Kutta methods is given by the following theorem.

Theorem 2.26. *Consider a DAE* (2.13) *with differentiation index $\nu_d = 1$ and consistent initial values $(\mathbf{x}_0, \mathbf{y}_0)$. Assume that $\frac{\partial}{\partial \mathbf{y}} \mathbf{g}_2(\mathbf{x}, \mathbf{y})$ has a bounded inverse in the neighborhood of the solution $(\mathbf{x}^*, \mathbf{y}^*)$. Consider, furthermore, an implicit Runge-Kutta method with classical order p that satisfies $C(q)$ with $p \geq q + 1$.*

- *The order of convergence for the $\mathbf{x}$-component is p.*
- *If the method is stiffly accurate, then the order of convergence for the $\mathbf{y}$-component is also p.*
- *If $-1 \leq R(\infty) < 1$, then the order of convergence for the $\mathbf{y}$-component is $q + 1$.*
- *If $R(\infty) = +1$, then the convergence order for the $\mathbf{y}$-component is q.*
- *If $|R(\infty)| > 1$, then the numerical solution diverges.*

Proof. See [35, 36, 50]. □

Theorem 2.26 shows that stiffly accurate Runge-Kutta methods not only compute an approximation to the analytical solution that is consistent. These methods also show the same convergence behavior for DAEs of the form (2.13) with differentiation index 1 as for ODEs. Hence, stiffly accurate Runge-Kutta methods like the Radau IIA methods are of special interest for the numerical solution of DAEs.

2.3.2. BDF methods

General linear multistep methods

As mentioned in the introduction to this section, general linear multistep methods applied to ODEs take the form

$$\sum_{j=0}^{r} \alpha_{r-j} \boldsymbol{\xi}_{i+1-j} = h \sum_{j=0}^{r} \beta_{r-j} \mathbf{f}(t_{i+1-j}, \boldsymbol{\xi}_{i+1-j}), \tag{2.32}$$

with coefficients $\alpha_j, \beta_j \in \mathbb{R}$, $j = 0, \ldots, r$ and $\alpha_r \neq 0$. If additionally $\alpha_0^2 + \beta_0^2 \neq 0$ holds, i.e. the method uses r steps to compute an approximation of the solution for the next step, then (2.32) is called a linear r-step method. Since we are able to scale the coefficients by multiplying (2.32) with an arbitrary scalar $\neq 0$, we may assume that either $\alpha_r = 1$ or $\beta_r = 1$. The method (2.32) is called a corrector method or an implicit method if $\beta_r \neq 0$. The method is an explicit or a predictor method if $\beta_r = 0$.

To examine the consistency, convergence and stability of linear multistep methods, we define the characteristic polynomial of a linear multistep method as follows.

Definition 2.27. (Characteristic polynomial of a linear multi-step method) The *characteristic polynomial* of a linear r-step method with coefficients $\alpha_0, \ldots, \alpha_r$ is given by

$$\rho(\lambda) = \alpha_r \lambda^r + \alpha_{r-1} \lambda^{r-1} + \cdots + \alpha_0. \tag{2.33}$$

□

Theorem 2.28. *A linear r-step method is consistent of order p if the coefficients $\alpha_j, \beta_j,\ j = 0, \dots, r$ satisfy*

$$\sum_{j=0}^{r} \alpha_j j^q = q \sum_{j=0}^{r} \beta_j j^{q-1},\ q = 0, \dots, p. \tag{2.34}$$

Proof. See [36] or [50]. □

Theorem 2.29. *The multistep method* (2.32) *is stable, if the roots of its characteristic polynomial* (2.33) *satisfy the following root or stability condition:*

- *The roots of* (2.33) *lie inside the unit disk.*
- *The roots of* (2.33) *with modulus 1 are simple roots.*

Proof. A proof of this theorem can be found in [50] or [73]. The proof in [73] uses the theory of difference equations, whereas the proof in [50] uses a more direct approach. □

With these two results, Theorem 2.18 now yields the convergence of order p of linear multistep methods with coefficients that satisfy (2.34) and the stability condition.

BDF methods

Backward difference formulae methods or BDF methods are linear r-step methods that are defined by the coefficients $\beta_0 = \dots = \beta_{r-1} = 0$ and $\beta_r = 1$. The remaining coefficients $\alpha_j,\ j = 0, \dots, r$ are chosen such that the method achieves a high order of consistency. With Theorem 2.28 and the convention that $0^0 = 1$, we get a system of linear equations

$$\underbrace{\begin{bmatrix} 0^0 & 1^0 & 2^0 & \dots & r^0 \\ 0^1 & 1^1 & 2^1 & \dots & r^1 \\ 0^2 & 1^2 & 2^2 & \dots & r^2 \\ & \vdots & \vdots & \ddots & \vdots \\ 0^p & 1^p & 2^p & \dots & r^p \end{bmatrix}}_{=:\boldsymbol{V}} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_r \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 2r \\ \vdots \\ pr^{p-1} \end{bmatrix}. \tag{2.35}$$

For $r = p$ the matrix $\boldsymbol{V}$ is just the $(r+1) \times (r+1)$ Vandermonde matrix which is nonsingular. Hence, the linear system (2.35) is uniquely solvable for $p = r$ and we will always be able to find coefficients $\alpha_j,\ j = 0, \dots, p$ such that the resulting BDF method is consistent of order p. Table 2.4 shows the coefficients for the BDF methods up to order 6.

In order to apply Theorem 2.18 we also need the stability of the BDF methods. Unfortunately, the next theorem shows that we will not be able to obtain BDF methods with arbitrary high order of convergence.

p	α_0	α_1	α_2	α_3	α_4	α_5	α_6
1	-1	1					
2	$\frac{1}{2}$	-2	$\frac{3}{2}$				
3	$-\frac{1}{3}$	$\frac{3}{2}$	-3	$\frac{11}{6}$			
4	$\frac{1}{4}$	$-\frac{4}{3}$	3	-4	$\frac{25}{12}$		
5	$-\frac{1}{5}$	$\frac{5}{4}$	$-\frac{10}{3}$	5	-5	$\frac{137}{60}$	
6	$\frac{1}{6}$	$-\frac{6}{5}$	$\frac{15}{4}$	$-\frac{20}{3}$	$\frac{15}{2}$	-6	$\frac{147}{60}$

Table 2.4.: Coefficients of BDF methods for $1 \leq p \leq 6$

Theorem 2.30. *An r-step BDF method is stable for $1 \leq r \leq 6$ and unstable for $r \geq 7$.*

Proof. A proof can be for example found in [36] or [37]. □

An easy and natural way to apply this method to a semi-explicit DAE is given in [15] by

$$\begin{aligned} \sum_{j=0}^{r} \alpha_{r-j} \boldsymbol{\xi}_{i+1-j} &= h\mathbf{f}_1(t_{i+1}, \boldsymbol{\xi}_{i+1}, \boldsymbol{\eta}_{i+1}) \\ 0 &= \mathbf{f}_2(t_{i+1}, \boldsymbol{\xi}_{i+1}, \boldsymbol{\eta}_{i+1}). \end{aligned} \tag{2.36}$$

Essentially this means that we require that the algebraic constraints are satisfied at each time step, i.e. the numerical solution is consistent for semi-explicit DAEs with differentiation index 1.

We have seen in Theorem 2.26, that the order of convergence may be reduced if we use an implicit Runge-Kutta method to discretize a semi-explicit DAE with differentiation index 1. As the next theorem shows this is not true for stable BDF methods.

Theorem 2.31. *Consider a BDF method of order p applied to a semi-explicit DAE* (2.43) *with differentiation index* 1 *as in* (2.36)*. If the roots of the characteristic polynomial* (2.33) *of the BDF method satisfy the stability condition, then the method is convergent of order p.*

Proof. The theorem follows from Theorem 5.26 in [50] and the fact that for BDF methods $\beta_0 = \ldots = \beta_{p-1} = 0$ and $\beta_p = 1$. □

2.4. General index reduction methods

DAEs with higher index are known to cause problems during the numerical integration with BDF methods or implicit Runge-Kutta methods, even though these methods work well for DAEs with differentiation index 1 and consistent initial values.

With this in mind, Section 2.4.1 will give a general overview of methods that reduce the index of a given DAE to 1. Section 2.4.2 then will discuss the index reduction by minimal extension which will be adapted to the special structural properties of DAEs from circuit simulation in Chapter 4.

2.4.1. Overview

Consider a quasi-linear DAE of the form

$$\begin{bmatrix} \mathbf{E}_1(t,\mathbf{x}_1,\mathbf{x}_2) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{x}}_1 \\ \dot{\mathbf{x}}_2 \end{bmatrix} + \begin{bmatrix} \mathbf{g}_1(t,\mathbf{x}_1,\mathbf{x}_2) \\ \mathbf{g}_2(t,\mathbf{x}_1,\mathbf{x}_2) \end{bmatrix} = \mathbf{0}, \tag{2.37}$$

with $\mathbf{E}_1(t,\mathbf{x}_1,\mathbf{x}_2)$ pointwise nonsingular. The easiest way to reduce the index of (2.37) by 1 would be to replace the constraints

$$\mathbf{g}_2(t,\mathbf{x}_1,\mathbf{x}_2) = \mathbf{0} \tag{2.38}$$

by their derivatives, obtaining the system

$$\begin{bmatrix} \mathbf{E}_1(t,\mathbf{x}_1,\mathbf{x}_2) & \mathbf{0} \\ \frac{\partial}{\partial \mathbf{x}_1}\mathbf{g}_2(t,\mathbf{x}_1,\mathbf{x}_2) & \frac{\partial}{\partial \mathbf{x}_2}\mathbf{g}_2(t,\mathbf{x}_1,\mathbf{x}_2) \end{bmatrix} \begin{bmatrix} \dot{\mathbf{x}}_1 \\ \dot{\mathbf{x}}_2 \end{bmatrix} + \begin{bmatrix} \mathbf{g}_1(t,\mathbf{x}_1,\mathbf{x}_2) \\ \frac{\partial}{\partial t}\mathbf{g}_2(t,\mathbf{x}_1,\mathbf{x}_2) \end{bmatrix} = \mathbf{0}. \tag{2.39}$$

However, this replacement would lead to the loss of information due to the differentiation. Hence, the numerical solution of (2.39) may not fulfill the original constraints (2.38). As these constraints often represent physical laws like conservation of mass or energy or Kirchhoff's Laws in case of DAEs from circuit simulation, the numerical solution of (2.39) may be physically meaningless.

Because of this, various suggestions have been made to conserve the original constraints while taking the derivatives into account. For example in [30], an approach is considered that transforms general nonlinear DAEs (2.1a) with differentiation index ν_d to ODEs (2.10) with help of the derivative array (2.8) of order ν_d. Under the assumption that the Jacobian of the hidden constraints (2.10b) $\frac{\partial}{\partial \mathbf{x}}\bar{\mathbf{f}}_2(t,\mathbf{x})$ has full row rank n_μ the hidden constraints are coupled to (2.10) by an additional Lagrangian multiplier $\boldsymbol{\mu}$ of size n_μ. This leads to the system

$$\begin{aligned} \dot{\mathbf{x}} &= \bar{\mathbf{f}}_1(t,\mathbf{x}) + \frac{\partial}{\partial \mathbf{x}}\bar{\mathbf{f}}_2(t,\mathbf{x})^T\boldsymbol{\mu} \\ \mathbf{0} &= \bar{\mathbf{f}}_2(t,\mathbf{x}). \end{aligned} \tag{2.40}$$

Theorem 2.32. *Consider a general nonlinear DAE* (2.1a) *with differentiation index* ≥ 2 *and the corresponding system* (2.40)*. Then,* (2.40) *has differentiation index* 2 *and for every solution* $(\mathbf{x}^*,\boldsymbol{\mu}^*)$ *of* (2.40) *it holds that* $\boldsymbol{\mu}^* = 0$ *and* $\mathbf{x}^*$ *is a solution of* (2.1a)*. Moreover, every solution* $\mathbf{x}^*$ *of* (2.1a) *together with* $\boldsymbol{\mu}^* = 0$ *is a solution of* (2.40).

Proof. Cf. Theorem 2.5.1 and Theorem 2.5.2 in [15]. □

Although the differentiation index of (2.40) still is equal to 2, the special structure of the system allows for a better numerical treatment of this system in comparison with the original system (2.1a).

Other approaches omit the introduction of the Lagrangian multiplier $\boldsymbol{\mu}$ and rather try to find a solution to the overdetermined system

$$\begin{aligned} \mathbf{0} &= \mathbf{f}(t, \mathbf{x}, \dot{\mathbf{x}}), \\ \mathbf{0} &= \bar{\mathbf{f}}_2(t, \mathbf{x}), \end{aligned} \tag{2.41}$$

cf. [13, 19]. All of these approaches are more or less based on the definition of the differentiation index.

If we consider a DAE (2.1a) that satisfies Hypothesis 2.12, then we are able to use the transformations $\mathbf{Z}_1$ and $\mathbf{Z}_2$ defined in the hypothesis to derive the equivalent strangeness-free formulation

$$\begin{aligned} \mathbf{0} &= \mathbf{Z}_1^T \mathbf{f}(t, \mathbf{x}, \dot{\mathbf{x}}), \\ \mathbf{0} &= \mathbf{Z}_2^T \mathbf{F}_{\nu_s}(t, \mathbf{x}, \dot{\mathbf{x}}, \ldots, \mathbf{x}^{(\nu_s+1)}). \end{aligned} \tag{2.42}$$

Note that due to the definition of $\mathbf{Z}_1$ and $\mathbf{Z}_2$ in Hypothesis 2.12, (2.42) only depends on t, $\mathbf{x}$ and $\dot{\mathbf{x}}$, but not on the higher derivatives of $\mathbf{x}$ [50].

2.4.2. Index reduction by minimal extension

The index reduction methods presented in the previous section are suitable for a wide range of nonlinear DAEs. However, they suffer from several drawbacks. As a start, the determination of the hidden constraints requires the computation of the derivative array which becomes computationally expensive especially for DAEs with high index. After the computation of the derivative array, the information about the hidden constraints has to be extracted. In case of the strangeness-free formulation (2.42) this is done by several rank-revealing singular value decompositions. If that information is finally available, then one of the index reduced systems (2.40), (2.41) or (2.42) has to be solved. In both cases (2.40) and (2.41), the index reduced system is larger than the original one which may add considerably to the computational costs if the original DAE is large and contains many hidden constraints.

In view of these issues, attempts to lower the computational cost were made. In [68] it is suggested to use only those derivatives that are actually needed to determine the hidden constraints instead of the full derivative array. Also, the Pantelides algorithm proposed in [68] to compute these derivatives is based on a combinatorial method rather than on algebraic methods to further lower the computational costs. Based on this approach, an index reduction method was proposed in [64]. Instead of explicitly computing the hidden constraints, the set of necessary derivatives which have been determined by the algorithm from [68] are added to the original problem. To avoid having to deal with the overdetermined system that arises, certain derivatives of the unknowns are replaced by new algebraic variables. The variables that have to be replaced are again determined by the Pantelides algorithm.

Unfortunately, there is no relation between the so-called structural index determined by the algorithm proposed in [68] and the differentiation index [40]. Moreover, the algorithm may not determine a minimal set of derivatives. However, in [48] the basic idea of the index reduction method introduced in [64] was used to reduce the differentiation index of DAEs with additional structural properties.

Since the index reduction by minimal extension actually needs a high amount of structural information about the considered DAE, we will not present this technique for general nonlinear DAEs. Instead, we will demonstrate the method by considering a quasi-linear DAE of the special form

$$\mathbf{0} = \dot{\mathbf{x}} + \mathbf{g}_1(t, \mathbf{x}, \mathbf{y}), \tag{2.43a}$$

$$\mathbf{0} = \mathbf{g}_2(t, \mathbf{x}), \tag{2.43b}$$

where $\mathbf{x} \in \mathbb{D}_\mathbf{x} \subseteq \mathbb{R}^{n_x}$, $\mathbf{y} \in \mathbb{D}_\mathbf{y} \subseteq \mathbb{R}^{n_y}$ and $\mathbf{g}_1 \in \mathcal{C}^1\left(\mathbb{I} \times \mathbb{D}_\mathbf{x} \times \mathbb{D}_\mathbf{y}, \mathbb{R}^{n_x}\right)$, $\mathbf{g}_2 \in \mathcal{C}^2\left(\mathbb{I} \times \mathbb{D}_\mathbf{x}, \mathbb{R}^{n_y}\right)$. In addition we assume again that (2.43) is regular. In this case, $\frac{\partial}{\partial \mathbf{x}}\mathbf{g}_2(t, \mathbf{x})$ has full row rank. This DAE has differentiation index 2 if $\frac{\partial}{\partial \mathbf{x}}\mathbf{g}_2 \frac{\partial}{\partial \mathbf{y}}\mathbf{g}_1$ is nonsingular. This is easy to see, since from the differentiation of (2.43b) we get

$$\mathbf{0} = \frac{\partial}{\partial \mathbf{x}}\mathbf{g}_2(t, \mathbf{x})\dot{\mathbf{x}} + \frac{\partial}{\partial t}\mathbf{g}_2(t, \mathbf{x}), \tag{2.44}$$

which together with (2.43a) leads to the hidden constraint

$$\mathbf{0} = -\frac{\partial}{\partial \mathbf{x}}\mathbf{g}_2(t, \mathbf{x})\mathbf{g}_1(t, \mathbf{x}, \mathbf{y}) + \frac{\partial}{\partial t}\mathbf{g}_2(t, \mathbf{x}). \tag{2.45}$$

Since $\frac{\partial}{\partial \mathbf{x}}\mathbf{g}_2 \frac{\partial}{\partial \mathbf{y}}\mathbf{g}_1$ is nonsingular, the differentiation of equation (2.45) finally leads to a differential equation for $\mathbf{y}$

$$\begin{aligned}\dot{\mathbf{y}} = -\left[\frac{\partial}{\partial \mathbf{x}}\mathbf{g}_2 \frac{\partial}{\partial \mathbf{y}}\mathbf{g}_1\right]^{-1} &\left[\frac{\partial}{\partial \mathbf{x}}\left[\frac{\partial}{\partial \mathbf{x}}\mathbf{g}_2(t, \mathbf{x})\mathbf{g}_1(t, \mathbf{x}, \mathbf{y})\right]\dot{\mathbf{x}} + \frac{\partial}{\partial t}\left[\frac{\partial}{\partial \mathbf{x}}\mathbf{g}_2(t, \mathbf{x})\mathbf{g}_1(t, \mathbf{x}, \mathbf{y})\right]\right. \\ &\left. -\frac{d}{dt}\frac{\partial}{\partial t}\mathbf{g}_2(t, \mathbf{x})\right].\end{aligned}$$

Hence the differentiation index is indeed equal to 2.

The derivative needed for the index reduction is given by (2.44) which is added to the original DAE system (2.43) to yield the overdetermined system

$$\mathbf{0} = \dot{\mathbf{x}} + \mathbf{g}_1(t, \mathbf{x}, \mathbf{y}), \tag{2.46a}$$

$$\mathbf{0} = \mathbf{g}_2(t, \mathbf{x}), \tag{2.46b}$$

$$\mathbf{0} = \frac{\partial}{\partial \mathbf{x}}\mathbf{g}_2(t, \mathbf{x})\dot{\mathbf{x}} + \frac{\partial}{\partial t}\mathbf{g}_2(t, \mathbf{x}). \tag{2.46c}$$

Now, the variables that need to be replaced have to be determined. To do so, we examine the derivative (2.46c) and split the variable $\mathbf{x}$ into $\mathbf{x}_1$ and $\mathbf{x}_2$ such that $\mathbf{g}_{2,\mathbf{x}_2}$

is nonsingular. This splitting is locally possible due to the regularity of (2.43). From equation (2.46c), we thus obtain

$$\mathbf{0} = \frac{\partial}{\partial \mathbf{x}_1}\mathbf{g}_2(t,\mathbf{x}_1,\mathbf{x}_2)\dot{\mathbf{x}}_1 + \frac{\partial}{\partial \mathbf{x}_2}\mathbf{g}_2(t,\mathbf{x}_1,\mathbf{x}_2)\dot{\mathbf{x}}_2 + \frac{\partial}{\partial t}\mathbf{g}_2(t,\mathbf{x}_1,\mathbf{x}_2). \tag{2.47}$$

The derivatives of $\mathbf{x}_2$ in (2.47) can be computed once the derivatives of $\mathbf{x}_1$ are given. Thus, $\dot{\mathbf{x}}_2$ acts more like an algebraic than a differential variable in (2.46). Hence, to make this fact visible in the extended system, $\dot{\mathbf{x}}_2$ is replaced by an algebraic variable $\mathbf{z}$. This is the crucial point of the index reduction by minimal extension. From a numerical point of view, the consequence of this replacement is that $\mathbf{x}_2$ will be excluded from discretization in a numerical integration method and hence not influenced by discretization errors. This guarantees that the hidden algebraic constraint will be fulfilled up to round-off and truncation errors if the system is numerically integrated with an appropriate method (cf. Section 2.3).

In order to be able to replace $\dot{\mathbf{x}}_2$ by $\mathbf{z}$ in (2.46), we also have to split (2.46a) into two equations

$$\mathbf{0} = \dot{\mathbf{x}}_1 + \mathbf{g}_{11}(t,\mathbf{x}_1,\mathbf{x}_2,\mathbf{y}), \tag{2.48a}$$
$$\mathbf{0} = \dot{\mathbf{x}}_2 + \mathbf{g}_{12}(t,\mathbf{x}_1,\mathbf{x}_2,\mathbf{y}) = \mathbf{z} + \mathbf{g}_{12}(t,\mathbf{x}_1,\mathbf{x}_2,\mathbf{y}). \tag{2.48b}$$

Note that the replacement yields a new algebraic equation (2.48b). The DAE obtained in this way from (2.46) is given by

$$\mathbf{0} = \dot{\mathbf{x}}_1 + \mathbf{g}_{11}(t,\mathbf{x}_1,\mathbf{x}_2,\mathbf{y}), \tag{2.49a}$$
$$\mathbf{0} = \mathbf{z} + \mathbf{g}_{12}(t,\mathbf{x}_1,\mathbf{x}_2,\mathbf{y}), \tag{2.49b}$$
$$\mathbf{0} = \mathbf{g}_2(t,\mathbf{x}_1,\mathbf{x}_2), \tag{2.49c}$$
$$\mathbf{0} = \frac{\partial}{\partial \mathbf{x}_1}\mathbf{g}_2(t,\mathbf{x}_1,\mathbf{x}_2)\dot{\mathbf{x}}_1 + \frac{\partial}{\partial \mathbf{x}_2}\mathbf{g}_2(t,\mathbf{x}_1,\mathbf{x}_2)\mathbf{z} + \frac{\partial}{\partial t}\mathbf{g}_2(t,\mathbf{x}_1,\mathbf{x}_2), \tag{2.49d}$$

where $\mathbf{x}_i \in \mathbb{D}_{\mathbf{x}_i} \subseteq \mathbb{R}^{n_{x_i}},\ i = 1,2$, $\mathbf{y} \in \mathbb{D}_{\mathbf{y}} \subseteq \mathbb{R}^{n_y}$ and $\mathbf{g}_{1i} \in \mathcal{C}^1\left(\mathbb{I} \times \mathbb{D}_{\mathbf{x}_1} \times \mathbb{D}_{\mathbf{x}_2} \times \mathbb{D}_{\mathbf{y}}, \mathbb{R}^{n_{x_i}}\right)$, $i = 1,2$, $\mathbf{g}_2 \in \mathcal{C}^2\left(\mathbb{I} \times \mathbb{D}_{\mathbf{x}_1} \times \mathbb{D}_{\mathbf{x}_2}, \mathbb{R}^{n_y}\right)$. The system (2.49) has differentiation index 1 since we only need to differentiate (2.49b) and (2.49c) once to get an expression for $\dot{\mathbf{z}}$ and $\dot{\mathbf{x}}_2$. An expression for $\mathbf{y}$ is obtained by inserting (2.49b) into (2.49d) and differentiating the resulting equation once.

2.5. Numerical software for DAEs

As we have seen in the previous sections of this chapter, the analytical as well as the numerical properties of DAEs may differ greatly from those of ODEs. Therefore not every method that is suitable for the numerical solution of ODEs is suitable for the treatment of DAEs. In Section 2.3, one-step methods and linear multistep methods have been discussed.

Among one-step methods, implicit Runge-Kutta methods of Radau IIA type are well suited for the numerical solution of DAEs (cf. Section 2.3.1). These methods are L-stable and stiffly accurate. The code Radau5 [8] uses the Radau IIA method of order 5 for the discretization of a quasi-linear DAE

$$\mathbf{E}\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}) \tag{2.50}$$

with constant leading term $\mathbf{E}$ [38]. It has been shown in [35] and [38] that the method converges for DAEs (2.50) with differentiation index 1 and 2. The method converges even for DAEs of differentiation index 3 if the DAEs are in a Hessenberg form

$$\begin{aligned}\dot{\mathbf{x}}_1 &= \mathbf{f}_1(t, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3),\\ \dot{\mathbf{x}}_2 &= \mathbf{f}_2(t, \mathbf{x}_1, \mathbf{x}_2),\\ \mathbf{0} &= \mathbf{f}_3(t, \mathbf{x}_2),\end{aligned}$$

where $\frac{\partial}{\partial \mathbf{x}_2}\mathbf{f}_3 \frac{\partial}{\partial \mathbf{x}_1}\mathbf{f}_2 \frac{\partial}{\partial \mathbf{x}_3}\mathbf{f}_1$ is nonsingular. A direct application of Radau5 to more general quasi-linear DAEs with non-constant leading term as in (2.4) is not possible, since the code exploits the special structure of (2.50) to split the linear system of size $3n$ that arises during the integration into two smaller systems of sizes n and $2n$. However, it is possible to transform (2.4) into a system of the form

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{y},\\ \mathbf{0} &= \mathbf{E}(t, \mathbf{x})\mathbf{y} + \mathbf{g}(t, \mathbf{x}),\end{aligned} \tag{2.51}$$

and apply Radau5 to (2.51) instead of the original system (2.4). One drawback of this approach is that the differentiation index of (2.51) is one higher than the differentiation index of (2.4) [15, 30]. Hence, Radau5 can be applied to general quasi-linear systems (2.4) of differentiation index up to 2. Another code that uses Radau IIA type methods is RadauP [9]. The available methods are of order 5, 9 and 13 [38] and are suitable for DAEs (2.50) up to differentiation index 3 or DAEs (2.4) up to differentiation index 2. Both Radau5 and RadauP include an efficient and flexible stepsize control.

Codes that use BDF methods include Dassl [3], Daspk [2] and Ida, which is part of the Sundials package [10]. BDF methods are rather easy to implement, even for general DAEs (2.1a). The nonlinear systems that arise during integration are of size n. However, the development of a step size control proves to be difficult and the convergence behavior of BDF methods with variable step size is not completely understood [15]. Moreover, BDF methods cannot achieve an arbitrary high order of convergence as methods of consistency order ≥ 7 become unstable (cf. Section 2.3.2).

All codes presented previously attempt to integrate a given DAE "as it is" and do not attempt to reduce the index of the DAE. Codes that include an index reduction technique are Gelda [4] for linear DAEs with variable coefficients [52]

$$\mathbf{E}(t)\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x} + \mathbf{f}(t),$$

Genda [5] for general nonlinear DAEs (2.1a) [55] and Geoms [6] for DAEs that arise in the simulation of multibody systems [72]. The index reduction in Gelda and Genda is based on Hypothesis 2.12. Instead of integrating the original DAE, the DAE is transformed to the equivalent strangeness-free form (2.42) which is then integrated with BDF methods. In GELDA it is also possible to use the Radau IIA method of order 5 as implemented in RADAU5. Both codes are available in the MATLAB [7] toolbox DAESOLVE [54].

3. DAEs in circuit simulation

The simulation of electrical circuits is an important part of the development of VLSI chips for various applications. The circuits involved may contain several millions of circuit elements. Hence a systematic way to generate the equations that describe such a circuit is essential. Since the behaviour of an electrical circuit is defined by the elements which it contains and the way in which the elements are connected with each other, graph theory, combined with Kirchhoff's Laws and the characteristic equations which describe the circuit elements offers such a systematic approach.

In this chapter, some basic definitions concerning the description of general oriented graphs will be given Section 3.1. Section 3.2 presents the Modified Nodal Analysis and the charge-oriented Modified Nodal Analysis as a systematic way to derive a DAE which models the behavior of a circuit from a graph theoretical description of the circuit. Section 3.3 presents the main results concerning the index determination for DAEs arising from both Modified Nodal Analysis and charge-oriented Modified Nodal Analysis.

3.1. Introduction to graph theory

Most of the definitions and results given in this section have been taken from [14, 23, 41, 75], where an in-depth treatment of general graph theory [14, 41] or graph theory applied to circuit simulation [23, 75] can be found. However, the notation used in the following has been adapted to the notation that is used in circuit simulation.

In addition to the fact that the notation in graph theory is often ambiguous there is also the problem of the correct graph theoretical description of circuits. On one hand, the currents through the elements as well as the voltages across the elements are directed quantities. Hence, their orientations should be considered in the graph theoretical description. However, since currents and voltages are allowed to be negative, the direction of the currents and voltages only determines the sign of these quantities but not the topology of the circuit itself. This implies that we are able to neglect the orientations when we are only interested in the topology and not in the actual values.

3.1.1. Basic definitions

Definition 3.1. (Oriented graph) An *oriented graph* $\mathfrak{G}$ is a pair $(\mathfrak{N}, \mathfrak{B})$, where $\mathfrak{N} = \{\mathfrak{n}_1 \dots \mathfrak{n}_N\}$ is a finite set and $\mathfrak{B}$ a set of ordered pairs of elements of $\mathfrak{N}$. The elements $\mathfrak{n}_k,\ k = 1, \dots N$ of $\mathfrak{N}$ are called *nodes* and the elements $\mathfrak{b}_{k_1,k_2} = <\mathfrak{n}_{k_1}, \mathfrak{n}_{k_2}>, \mathfrak{n}_{k_1} \neq \mathfrak{n}_{k_2}, \mathfrak{n}_{k_1}, \mathfrak{n}_{k_2} \in \mathfrak{N}$ are called *branches* of $\mathfrak{G}$. The number of branches $|\mathfrak{B}|$ will be denoted by B. The graph $\mathfrak{G}$ is also denoted by $\mathfrak{G}(\mathfrak{N}, \mathfrak{B})$. □

Note that in an oriented graph a branch $\mathfrak{b}_{k_1,k_2}$ means that there is a connection leading from $\mathfrak{n}_{k_1}$ to $\mathfrak{n}_{k_2}$. However, the connection cannot be used in the other way. If we drop this restriction and define the branches of a graph such that the order of the nodes does not matter, then we obtain a *non-oriented graph.* In general, non-oriented graphs are referred to as graphs. Note that for most definitions and results presented in this section there also exist analogous definitions and results for graphs, cf. [41].

Definition 3.2. (Incidence relation) Consider an oriented graph $\mathfrak{G}$. The branch $\mathfrak{b}_{k_1,k_2} = < \mathfrak{n}_{k_1}, \mathfrak{n}_{k_2} >$ is *incident* with the nodes $\mathfrak{n}_{k_1}$ and $\mathfrak{n}_{k_2}$. The branch $\mathfrak{b}_{k_1,k_2}$ leaves node $\mathfrak{n}_{k_1}$ and enters node $\mathfrak{n}_{k_2}$. □

If a node is not incident with any branch, then the node is called an *isolated node.* If for each pair of nodes there is at most one branch which is incident with the nodes, then the graph is *simple.* If multiple branches, i.e. sets of branches that are incident with the same pair of nodes, are allowed, then $\mathfrak{G}$ is called a *multigraph.* In this section, we will only consider simple graphs.

Definition 3.3. (Degree of a node) The *degree* $d(\mathfrak{n}_k) = d_k$ of a node $\mathfrak{n}_k$ is the number of branches that are incident with $\mathfrak{n}_k$. □

Lemma 3.4. *In any graph the number of nodes with odd degree is even.*

Proof. Since any branch in a graph is incident with exactly two nodes, each branch is counted exactly two time when summing up the degrees of the nodes, hence

$$\sum_{k=1}^{N} d_k = 2B. \tag{3.1}$$

Since the righthand side of (3.1) is even, the number of odd terms d_k in the sum must be even, too. □

Definition 3.5. (Path, simple path) A *path of length p* between two nodes $\mathfrak{n}_{j_0}$, $\mathfrak{n}_{j_p}$ of a graph $\mathfrak{G}$ is a sequence of nodes $\mathfrak{n}_{j_0}, \dots, \mathfrak{n}_{j_p}$ such that for every two consecutive nodes $\mathfrak{n}_{j_{k-1}} \neq \mathfrak{n}_{j_k}$, $k = 1, \dots, p$ in this sequence either $\mathfrak{b}_{j_{k-1},j_k} \in \mathfrak{B}$ or $\mathfrak{b}_{j_k,j_{k-1}} \in \mathfrak{B}$. If in addition, $\mathfrak{n}_{j_k} \neq \mathfrak{n}_{j_l}$, $k, l \in \{0, \dots, p-1\}$, $k \neq l$, then the path is *simple.* □

Consider a path $\mathfrak{n}_{j_0}, \dots, \mathfrak{n}_{j_p}$ in an oriented graph $\mathfrak{G}$. Denote by $\mathfrak{b}_{j_k}$ the branch that connects node $\mathfrak{n}_{j_{k-1}}$ and node $\mathfrak{n}_{j_k}$, $k = 1, \dots, p$. If $\mathfrak{b}_{j_k} = \mathfrak{b}_{j_{k-1},j_k}$, the branch is called a *forward branch* of the path. If, conversely, $\mathfrak{b}_{j_k} = \mathfrak{b}_{j_k,j_{k-1}}$, then the branch is called a *backward branch* of the path.

Definition 3.6. (Connected graph) An oriented graph $\mathfrak{G}$ is *connected* if for every pair of nodes there exists a path between them. □

Definition 3.7. (Subgraph, isolated node) Consider a connected graph $\mathfrak{G} = \mathfrak{G}(\mathfrak{N}, \mathfrak{B})$. A *subgraph* $\widetilde{\mathfrak{G}} = \widetilde{\mathfrak{G}}(\widetilde{\mathfrak{N}}, \widetilde{\mathfrak{B}})$ of $\mathfrak{G}$ is a graph such that $\widetilde{\mathfrak{N}} \subseteq \mathfrak{N}$, $\widetilde{\mathfrak{B}} \subseteq \mathfrak{B}$ and $\widetilde{\mathfrak{b}} =< \widetilde{\mathfrak{n}}_1, \widetilde{\mathfrak{n}}_2 >$ with $\widetilde{\mathfrak{n}}_1, \widetilde{\mathfrak{n}}_2 \in \widetilde{\mathfrak{N}}$ for all branches $\widetilde{\mathfrak{b}} \in \widetilde{\mathfrak{B}}$. A connected component which consists of only one node is called *isolated node.* □

If $\mathfrak{G}$ is not connected, it consists of at least two separate connected subgraphs. These subgraphs are usually called *components* of $\mathfrak{G}$.

Lemma 3.8. *A connected graph with N nodes has at least $N-1$ branches.*

Proof. A proof of this lemma can be found in [41]. □

Definition 3.9. (Loop, cutset) Let $\mathfrak{G}$ be an oriented graph. A *loop* is a simple path such that $\mathfrak{n}_{j_0} = \mathfrak{n}_{j_p}$. A *cutset* is a set $\mathfrak{B}_c$ of branches of $\mathfrak{G}$ such that the graph $\mathfrak{G}_c$ that results when the branches in $\mathfrak{B}_c$ are deleted from $\mathfrak{G}$ has one more component then $\mathfrak{G}$, but adding any branch in $\mathfrak{B}_c$ to $\mathfrak{G}_c$ would result in a graph with the same number of components as $\mathfrak{G}$. □

A branch in a loop is said to be oriented in the same way as the loop if it is a forward branch of the loop. To define the orientation of a branch in a cutset $\mathfrak{B}_c$, we consider a connected component $\widetilde{\mathfrak{G}} = \widetilde{\mathfrak{G}}(\widetilde{\mathfrak{N}}, \widetilde{\mathfrak{B}})$ of an oriented graph and a cutset $\mathfrak{B}_c \subset \widetilde{\mathfrak{B}}$. By removing the branches in $\mathfrak{B}_c$, the node set $\widetilde{\mathfrak{N}}$ of $\widetilde{\mathfrak{G}}$ is split into $\widetilde{\mathfrak{N}}_1$ and $\widetilde{\mathfrak{N}}_2$ with $\widetilde{\mathfrak{N}}_1, \widetilde{\mathfrak{N}}_2 \subset \widetilde{\mathfrak{N}}$, $\widetilde{\mathfrak{N}}_1 \cap \widetilde{\mathfrak{N}}_2 = \emptyset$. A branch $\mathfrak{b}$ that belongs to $\mathfrak{B}_c$ is said to have the same orientation as the cutset if $\mathfrak{b} =< \mathfrak{n}_1, \mathfrak{n}_2 >$ with $\mathfrak{n}_1 \in \widetilde{\mathfrak{N}}_1$ and $\mathfrak{n}_2 \in \widetilde{\mathfrak{N}}_2$, otherwise it has opposite orientation.

Lemma 3.10. *A graph with N nodes that does not contain loops has at most $N-1$ branches.*

Proof. A proof of this lemma can be found in [41]. □

Definition 3.11. (Tree, forest) Consider a connected graph $\mathfrak{G} = \mathfrak{G}(\mathfrak{N}, \mathfrak{B})$. A *tree* $\mathfrak{T} = \mathfrak{T}(\mathfrak{N}_T, \mathfrak{B}_T)$ of $\mathfrak{G}$ is a subgraph with $\mathfrak{N}_T = \mathfrak{N}$ which is connected but does not contain any loops. If $\mathfrak{G}$ is not connected and consists of F components, then it is possible to find a tree $\mathfrak{T}_j = \mathfrak{T}_j(\mathfrak{N}_{T_j}, \mathfrak{B}_{T_j}), j = 1, \dots, F$ in each of the components of $\mathfrak{G}$. These trees form a *forest* which will also be denoted by $\mathfrak{T} = \mathfrak{T}(\mathfrak{N}_T, \mathfrak{B}_T)$. In this case, $\mathfrak{N}_T$ and $\mathfrak{B}_T$ are given by

$$\begin{aligned}\mathfrak{N}_T &= \mathfrak{N}_{T_1} \cup \dots \cup \mathfrak{N}_{T_F},\\ \mathfrak{B}_T &= \mathfrak{B}_{T_1} \cup \dots \cup \mathfrak{B}_{T_F}.\end{aligned}$$

□

For a given tree $\mathfrak{T}$ in $\mathfrak{G}$, the branches $\widetilde{\mathfrak{b}}_1, \dots, \widetilde{\mathfrak{b}}_{B_t}$ of $\mathfrak{T}$ are referred to as *tree branches* where B_t denotes the number of tree branches. Those branches $\widehat{\mathfrak{b}}_1, \dots, \widehat{\mathfrak{b}}_{B_c}$ of $\mathfrak{G}$ that are not contained in $\mathfrak{T}$ are called *connecting branches.* Here, B_c denotes the number of connecting branches.

It is possible to construct a tree $\mathfrak{T}$ in every connected graph $\mathfrak{G}$. For details on the algorithms see for example [14, 41].

Theorem 3.12. *A tree $\mathfrak{T}$ with N nodes has exactly $N-1$ branches.*

Proof. Since $\mathfrak{T}$ is defined as a connected graph that does not contain loops, Lemma 3.8 and Lemma 3.10 imply that $\mathfrak{T}$ has exactly $N-1$ branches. □

Remark 3.13. *If $\mathfrak{T}$ is a forest with N nodes and F components, then $\mathfrak{T}$ has exactly $N-F$ branches. The statement can be proved by considering each component separately and summing up the number of branches in each component.*

Lemma 3.14. *A tree $\mathfrak{T}$ with N nodes, $N \geq 2$ contains at least two nodes with degree 1.*

Proof. By Lemma 3.12, a tree $\mathfrak{T}$ with $N \geq 2$ nodes contains $N-1$ branches. Since $\mathfrak{T}$ is connected, $d_k \geq 1$, $k = 1, \ldots, N$. Assume that there is only one node $\mathfrak{n}$ with $d(\mathfrak{n}) = 1$. Then, the remaining $N-1$ branches have degree ≥ 2 and summing up the degrees of all nodes yields

$$\sum_{k=1}^{N} d_k \geq 1 + 2(N-1), \tag{3.2}$$

which contradicts (3.1). □

Corollary 3.15. *A graph in which every node has degree ≥ 2 is not a tree.*

For the remainder of this section, we will only consider non-oriented graphs. Again, there are similar results for oriented graphs. However, we will only need the results as they are presented in the following.

Definition 3.16. (Complete graph) A *complete graph* with N nodes is a connected graph in which every node has degree $N-1$. □

Definition 3.17. (Connectivity, k-connected graph) The *connectivity* $\kappa(\mathfrak{G})$ of a graph $\mathfrak{G} = \mathfrak{G}(\mathfrak{N}, \mathfrak{B})$ is defined as follows. If $\mathfrak{G}$ is a complete graph, then $\kappa(\mathfrak{G}) = N-1$. Otherwise, $\kappa(\mathfrak{G})$ is the number of elements of the smallest subset $\mathfrak{N}^*$ of $\mathfrak{N}$ such that $\mathfrak{G}^*(\mathfrak{N} \setminus \mathfrak{N}^*, \mathfrak{B} \setminus \mathfrak{B}^*)$ is not connected. Here, $\mathfrak{B}^*$ denotes the subset of $\mathfrak{B}$ that only contains the branches that are incident with nodes in $\mathfrak{N}^*$.

A graph $\mathfrak{G}$ is called *k-connected* if $\kappa(\mathfrak{G}) \geq k$. □

Example 3.18. Figure 3.1 shows some graphs with connectivity 1. The possible choices for the subset $\mathfrak{N}^*$ are marked in the figure by a circle around the nodes. □

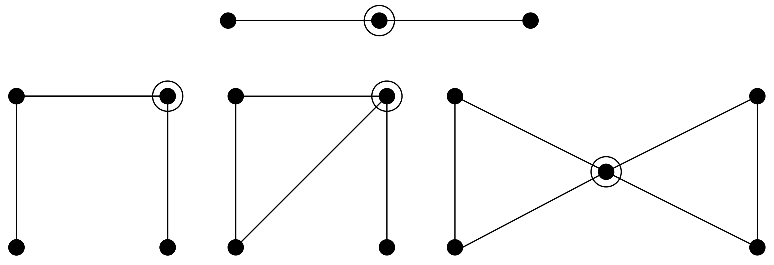


Figure 3.1.: Graphs with $\kappa(\mathfrak{G}) = 1$

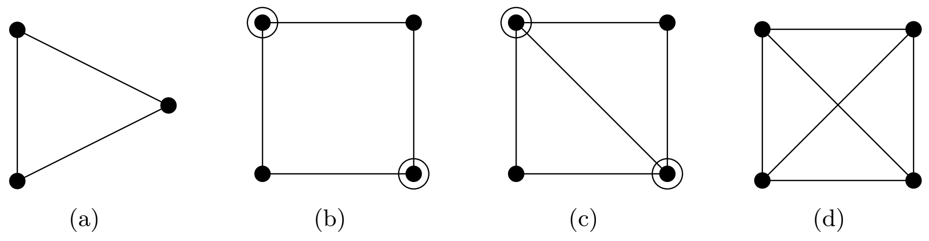


Figure 3.2.: 2-connected graphs

Example 3.19. Figure 3.2 shows some 2-connected graphs. The connectivity of 3.2(a) is 2, since it is a complete graph with three nodes. The connectivity of the graphs 3.2(b) and 3.2(c) is also 2. For both graphs, the nodes that disconnect the graphs are marked by circles. The connectivity for 3.2(d) is 3, since it is a complete graph with four nodes. Hence, by Definition 3.17, the graph 3.2(d) is 2-connected as well as 3-connected. □

Lemma 3.20. *Let $\mathfrak{G}$ be a graph with at least three nodes and with no isolated nodes. Then the following conditions are equivalent:*

1. *$\mathfrak{G}$ is 2-connected.*
2. *For every pair of nodes of $\mathfrak{G}$, there exists a loop containing both of them.*
3. *For each node $\mathfrak{n}$ and every branch $\mathfrak{b}$ of $\mathfrak{G}$, there exists a loop containing both $\mathfrak{n}$ and $\mathfrak{b}$.*
4. *For every pair of branches of $\mathfrak{G}$, there exists a loop containing both of them.*
5. *For every pair of nodes $\mathfrak{n}_1, \mathfrak{n}_2$ and every branch $\mathfrak{b}$ of $\mathfrak{G}$, there exists a path from $\mathfrak{n}_1$ to $\mathfrak{n}_2$ containing $\mathfrak{b}$.*
6. *For every triple of nodes $\mathfrak{n}_1, \mathfrak{n}_2, \mathfrak{n}_3$ of $\mathfrak{G}$ there exists a path from $\mathfrak{n}_1$ to $\mathfrak{n}_2$ containing $\mathfrak{n}_3$.*
7. *For every triple of nodes $\mathfrak{n}_1, \mathfrak{n}_2, \mathfrak{n}_3$ of $\mathfrak{G}$ there exists a path from $\mathfrak{n}_1$ to $\mathfrak{n}_2$ not containing $\mathfrak{n}_3$.*

Proof. A proof of this lemma can be found in [41]. A slightly different version of this lemma with proof can also be found in [71]. □

Definition 3.21. (2-connected component, block) A subgraph of $\mathfrak{G}$ that is 2-connected is called a 2-*connected component* or a *block*. □

Remark 3.22. *Although we have restricted the discussion to simple graphs, the definitions and results given throughout this section can be extended to multigraphs by considering the subgraphs that result from removing multiple branches in multigraphs.*

3.1.2. Graph related matrices

In this section we will not restrict the discussion to simple graphs. Rather, we will include multigraphs as the network graphs of circuits are multigraphs in general.

The incidence relation defined in Definition 3.2 may be used to fully describe $\mathfrak{G}$.

Definition 3.23. (Incidence matrix) Consider an oriented graph $\mathfrak{G}$ and let the branches be numbered arbitrarily, e.g. $\mathfrak{B} = \{\mathfrak{b}_1, \dots, \mathfrak{b}_B\}$. The coefficients of the *incidence matrix* $\widetilde{\mathbf{A}} \in \mathbb{R}^{N \times B}$ of $\mathfrak{G}$ are given by

$$a_{kl} = \begin{cases} 1, & \text{if branch } \mathfrak{b}_l \text{ leaves node } \mathfrak{n}_k, \\ -1, & \text{if branch } \mathfrak{b}_l \text{ enters node } \mathfrak{n}_k, \\ 0, & \text{if branch } \mathfrak{b}_l \text{ is not incident with node } \mathfrak{n}_k. \end{cases}$$

□

Lemma 3.24. *A graph $\mathfrak{G}$ with incidence matrix $\widetilde{\mathbf{A}}$ is a forest if and only if the columns of $\widetilde{\mathbf{A}}$ are linearly independent.*

Proof. The proof will be given by showing that $\mathfrak{G}$ contains a loop if and only if the columns of $\widetilde{\mathbf{A}}$ are linearly dependent.

Assume that $\mathfrak{G}$ contains the loop $\mathfrak{n}_{j_0}, \dots \mathfrak{n}_{j_p}$ with the branches $\mathfrak{b}_{j_k}$, $k = 1, \dots, p$ connecting the nodes in the loop. Let $\widetilde{\mathbf{a}}_{j_k}$, $k = 1, \dots, p$ be the columns of $\widetilde{\mathbf{A}}$ that correspond to those branches and set

$$\alpha_k = \begin{cases} 1, & \text{if } \mathfrak{b}_{j_k} \text{ is a forward branch,} \\ -1, & \text{if } \mathfrak{b}_{j_k} \text{ is a backward branch,} \end{cases} \tag{3.3}$$

$k = 1, \dots, p$. Then $\alpha_1 \widetilde{\mathbf{a}}_{j_1} + \dots + \alpha_k \widetilde{\mathbf{a}}_{j_p} = 0$, since each node that belongs to a loop is incident with exactly two branches in the loop and the signs of the corresponding entries in $\widetilde{\mathbf{A}}$ depend on whether the branches are forward or backward branches.

On the other hand, assume that the columns of $\widetilde{\mathbf{A}}$ are linearly dependent. Then, there exists a subset of columns $\widetilde{\mathbf{a}}_{j_1}, \dots, \widetilde{\mathbf{a}}_{j_p}$ and integers $\alpha_k \neq 0$, $k = 1, \dots, p$ such that $\alpha_1 \widetilde{\mathbf{a}}_{j_1} + \dots + \alpha_p \widetilde{\mathbf{a}}_{j_p} = 0$. Let $\mathfrak{B}'$ be the set of branches that correspond to the columns $\widetilde{\mathbf{a}}_{j_k}$, $k = 1, \dots, p$ and $\mathfrak{N}'$ the set of nodes that is incident with the branches in $\mathfrak{B}'$. Due to the fact that the columns are linearly dependent, every node in $\mathfrak{N}'$ has at least degree 2. Then, Corollary 3.15 shows that $\mathfrak{G}'(\mathfrak{N}', \mathfrak{B}')$ is not a tree and hence $\mathfrak{G}$ is not a forest. □

Lemma 3.25. *Let $\mathfrak{G}$ be a connected graph with N nodes. Then* rank $\widetilde{\mathbf{A}} = N - 1$.

Proof. Since every branch in $\mathfrak{G}$ is incident with exactly two nodes, there are exactly two nonzero entries in each column of $\widetilde{\mathbf{A}}$, namely one equal to $+1$ and one equal to -1. Hence, summing up the rows of $\widetilde{\mathbf{A}}$ yields a row with all entries equal to zero and therefore rank $\widetilde{\mathbf{A}} \leq N - 1$.

On the other hand it is possible to construct a tree $\mathfrak{T}$ in $\mathfrak{G}$. By Lemma 3.12, $\mathfrak{T}$ has $N - 1$ branches. Consider the incidence matrix $\widetilde{\mathbf{A}}_T \in \mathbb{R}^{N \times (N-1)}$ of $\mathfrak{T}$. This matrix is a sub-matrix of $\widetilde{\mathbf{A}}$ and is made up of the columns that correspond to the tree branches $\widetilde{\mathfrak{b}}_1, \dots, \widetilde{\mathfrak{b}}_{B_t}$. Due to Lemma 3.24, rank $\widetilde{\mathbf{A}}_T = N-1$ and hence rank $\widetilde{\mathbf{A}} \geq N-1$. Therefore, rank $\widetilde{\mathbf{A}} = N-1$. □

Remark 3.26. *If $\mathfrak{G}$ is not connected but consists of F components, then* rank $\widetilde{\mathbf{A}} = N-F$*. As in Remark 3.13, this follows if Lemma 3.25 is applied to the components of $\mathfrak{G}$.*

If the incidence matrix of a connected graph is formed considering all branches of the graph and all its nodes except for a reference node, this results in a reduced incidence matrix $\mathbf{A}$ which has full row rank.

In view of Section 2.2, the kernels of $\mathbf{A}$ and $\mathbf{A}^T$ are important. Lemma 3.24 gives a characterization of the kernel of the reduced incidence matrix $\mathbf{A}$ which is stated in the following corollary.

Corollary 3.27. *Let $\mathfrak{G}$ be an oriented graph with incidence matrix $\mathbf{A}$ and $\mathfrak{m}$ an arbitrary loop in $\mathfrak{G}$. Define the row vector $\mathbf{m} = (m_k)_{k=1,\dots,B}$ in analogy to* (3.3) *as follows.*

$$m_k = \begin{cases} 1, & \textit{if branch } \mathfrak{b}_k \textit{ is a forward branch in the loop } \mathfrak{m}, \\ -1, & \textit{if branch } \mathfrak{b}_k \textit{ is a backward branch in the loop } \mathfrak{m} \\ 0, & \textit{if branch } \mathfrak{b}_k \textit{ does not belong to the loop } \mathfrak{m}, \end{cases} \tag{3.4}$$

$k = 1, \dots, B$. Then, by Lemma 3.24

$$\mathbf{m}^T \in \ker \mathbf{A}.$$

Moreover the kernel of $\mathbf{A}$ is exactly spanned by the vectors $\mathbf{m}^T$ that are defined by the loops in $\mathfrak{G}$.

Remark 3.28. *In the course of the discussion of graph theoretical properties of network graphs we may encounter branches that are incident with the same node at both ends. Such branches are called* self loops *and are not included in Definition 3.23. However, the definition can be extended to include self loops by setting the column of $\widetilde{\mathbf{A}}$ that corresponds to a self loop to zero.*

Since a self loop consists of just one branch which corresponds to a zero column in the incidence matrix $\mathbf{A}$, Lemma 3.27 can be extended to graphs with self loops in a straight forward way.

Later in this section, we will consider a set of loops that defines a basis of $\ker \mathbf{A}$, but first we will give a characterization of the kernel of $\mathbf{A}^T$. To do so, we first consider the complete incidence matrix $\widetilde{\mathbf{A}}$. By Remark 3.26, $\widetilde{\mathbf{A}}$ has rank $N-F$ if the corresponding graph $\mathfrak{G}$ has N nodes and consists of F components. Define the coefficients of the matrix $\widetilde{\mathbf{Z}} =\in \mathbb{R}^{N \times F}$ by

$$\widetilde{z}_{kl} = \begin{cases} 1, & \text{if node } \mathfrak{n}_k \text{ belongs to component } l, \\ 0, & \text{else.} \end{cases} \tag{3.5}$$

Since every node in $\mathfrak{G}$ belongs exactly to one component, there is only one nonzero element in each row of $\widetilde{\mathbf{Z}}$ and the columns of $\widetilde{\mathbf{Z}}$ are linearly independent. Moreover, since $\widetilde{\mathbf{Z}}^T\widetilde{\mathbf{A}}$ sums up the rows of $\widetilde{\mathbf{A}}$, we have

$$\widetilde{\mathbf{A}}^T\widetilde{\mathbf{Z}} = \mathbf{0}_{B\times F},$$

where $\mathbf{0}_{B\times F}$ denotes the zero matrix in $\mathbb{R}^{B\times F}$. Hence the columns of $\widetilde{\mathbf{Z}}$ form a basis of $\ker \widetilde{\mathbf{A}}^T$. To obtain a basis of $\ker \mathbf{A}^T$, we have to remove the column of $\widetilde{\mathbf{Z}}$ that corresponds to the component of $\mathfrak{G}$ that contains the reference node and the row of $\widetilde{\mathbf{Z}}$ that corresponds to the reference node. The resulting matrix will be denoted by $\mathbf{Z} \in \mathbb{R}^{(N-1)\times(F-1)}$. Note that we allow $\mathbf{Z} \in \mathbb{R}^{(N-1)\times 0}$. The following lemma summarizes this result.

Lemma 3.29. *Let $\mathfrak{G}$ be a graph and $\mathbf{A}$ its incidence matrix. Define $\mathbf{Z}$ by* (3.5) *for all nodes except the reference node and for all components except the one that includes the reference node. Then the columns of $\mathbf{Z}$ form a basis of* $\ker \mathbf{A}^T$.

Remark 3.30. *It is possible to interpret the product $\mathbf{Z}^T\mathbf{A}$ as the graph $\mathfrak{G}_Z$ which is obtained from $\mathfrak{G}$ by replacing each connected component by a single node. The branches of $\mathfrak{G}$ are transformed into self loops of $\mathfrak{G}_Z$. Hence, using the extended definition of the incidence matrix as introduced in Remark 3.28, the incidence matrix of $\mathfrak{G}_Z$ is a zero matrix. We will call this transformation a* contraction. *Note that this definition is different from the definitions given in for example [14, 41] since we allow the contracted graph to have self loops.*

Even though the incidence matrix is the most common way to describe the topology of a graph in circuit simulation, it is not the only one. There are two other frequently used matrices that represent a graph, namely the loop matrix and the cutset matrix. The following definitions will allow for a description of a directed graph in terms of a minimal set of loops or cutsets.

Definition 3.31. (Fundamental loop, fundamental cutset) Let $\mathfrak{G}$ be a connected graph and $\mathfrak{T}$ a tree of $\mathfrak{G}$. Then

1. every connecting branch closes a unique loop that consists of that connecting branch and tree branches only. These loops are called *fundamental loops.*
2. every tree branch defines a unique cutset that consists of that tree branch and connecting branches only. These cutsets are called *fundamental cutsets.*

□

A consequence of Definition 3.31 is the fact that there are exactly B_c fundamental loops and B_t fundamental cutsets in a graph $\mathfrak{G}$ where B_c is the number of connecting branches and B_t is the number of tree branches in $\mathfrak{G}$. Note that both fundamental loops and fundamental cutsets depend on the choice of the tree $\mathfrak{T}$ in $\mathfrak{G}$. However, as will be shown in the following, once a tree $\mathfrak{T}$ is chosen and the sets of fundamental loops and fundamental cutsets have been determined it is possible to express every other loop or cutset in $\mathfrak{G}$ in terms of fundamental loops or cutsets.

Definition 3.32. (Loop matrix) Let $\mathfrak{G}$ be an oriented graph and $\mathfrak{T}$ a forest in $\mathfrak{G}$. Consider all fundamental loops $\mathfrak{m}_j$, $j = 1, \dots, B_c$ in $\mathfrak{G}$ that are defined by $\mathfrak{T}$. Let the orientation of the loops be defined such that the connecting branches $\widehat{b}_j$, $j = 1, \dots, B_c$ are forward branches in the loops that they define. The coefficients of the *(fundamental) loop matrix* $\mathbf{M} \in \mathbb{R}^{B_c \times B}$ are given by

$$m_{kl} = \begin{cases} 1, & \text{if branch } \mathfrak{b}_l \text{ belongs to loop } \mathfrak{m}_k \text{ and is a forward branch,} \\ -1, & \text{if branch } \mathfrak{b}_l \text{ belongs to loop } \mathfrak{m}_k \text{ and is a backward branch,} \\ 0, & \text{if branch } \mathfrak{b}_l \text{ does not belong to loop } \mathfrak{m}_k. \end{cases} \tag{3.6}$$

□

Definition 3.33. (Cutset matrix) Let $\mathfrak{G}$ be an oriented graph and $\mathfrak{T}$ a forest in $\mathfrak{G}$. Consider all fundamental cutsets $\mathfrak{s}_i$, $i = 1, \dots, B_t$ in $\mathfrak{G}$ that are defined by $\mathfrak{T}$. Let the orientation of the cutsets be defined such that the tree branches $\widetilde{\mathfrak{b}}_i$, $i = 1, \dots, B_t$ have the same orientation as the cutsets they define. The coefficients of the *(fundamental) cutset matrix* $\mathbf{S} \in \mathbb{R}^{B_t \times B}$ are then given by

$$s_{kl} = \begin{cases} 1, & \text{if branch } \mathfrak{b}_l \text{ belongs to cutset } \mathfrak{s}_k \text{ and is oriented in the same way,} \\ -1, & \text{if branch } \mathfrak{b}_l \text{ belongs to cutset } \mathfrak{s}_k \text{ and is oriented in the opposite way,} \\ 0, & \text{if branch } \mathfrak{b}_l \text{ does not belong to cutset } \mathfrak{s}_k. \end{cases} \tag{3.7}$$

□

Lemma 3.34. *Let $\mathfrak{G}$ be an oriented graph and $\mathfrak{T}$ a forest of $\mathfrak{G}$. If the branches of $\mathfrak{G}$ are ordered such that $\mathfrak{B} = \{\widehat{\mathfrak{b}}_1, \dots, \widehat{\mathfrak{b}}_{B_c}, \widetilde{\mathfrak{b}}_1, \dots, \widetilde{\mathfrak{b}}_{B_t}\}$ then*

$$\mathbf{M} = \begin{bmatrix} \mathbf{I}_{B_c} & \mathbf{G} \end{bmatrix}, \tag{3.8a}$$

$$\mathbf{S} = \begin{bmatrix} -\mathbf{G}^T & \mathbf{I}_{B_t} \end{bmatrix}, \tag{3.8b}$$

where $\mathbf{I}_k$ denotes the $k \times k$ identity matrix and $\mathbf{G} \in \mathbb{R}^{B_c \times B_t}$.

Proof. A proof of the lemma can be found in [23, 75]. □

Remark 3.35. 1. *The definitions for the coefficients m_{kl} and s_{kl} can be used for any set of loops or cutsets. However the resulting loop or cutset matrix does not have to have full row rank, whereas the loop and cutset matrices defined by fundamental loops and cutsets do have full row rank.*

2. *Fundamental loops and fundamental cutsets are not the only sets of loops and cutsets that define full rank matrices. It is for example possible to determine a set of loops in such a way that*

$$\mathbf{M} = \begin{bmatrix} \widetilde{\mathbf{M}}_{B_c} & \widetilde{\mathbf{M}}_{B_t} \end{bmatrix}, \tag{3.9}$$

where $\widetilde{\mathbf{M}}_{B_c}$ is a triangular matrix [57, 77].

Lemma 3.36. *Let $\mathfrak{G}$ be an oriented graph with incidence matrix* $\mathbf{A}$ *and* $\mathfrak{T}$ *a forest in* $\mathfrak{G}$*. Let* $\mathfrak{m}_1, \dots, \mathfrak{m}_{B_c}$ *be the fundamental loops in* $\mathfrak{G}$ *that are defined by* $\mathfrak{T}$ *and let* $\mathbf{m}_1, \dots, \mathbf{m}_{B_c}$ *be the corresponding rows of the loop matrix* $\mathbf{M}$*. Then, for every loop in* $\mathfrak{G}$ *the vector* $\mathbf{m}$ *defined by* (3.4) *is a linear combination of* $\mathbf{m}_1, \dots, \mathbf{m}_{B_c}$*.*

Proof. From Lemma 3.34, it is obvious that the loop matrix $\mathbf{M}$ has full rank. Hence the rows $\mathbf{m}_1, \dots, \mathbf{m}_{B_c}$ are linearly independent.

Let $\widetilde{\mathfrak{m}}$ be an arbitrary non-fundamental loop and define the vector $\widetilde{\mathbf{m}}$ by (3.4). Without loss of generality the branches are ordered as in Lemma 3.34. Consider the matrix

$$\mathbf{M}_+ = \begin{bmatrix} \mathbf{M} \\ \hline \widetilde{\mathbf{m}} \end{bmatrix}$$

and assume that $\mathbf{M}_+$ has full row rank $B_c + 1$. Then, there exists an equivalence transformation such that

$$\mathbf{M}_+ \sim \left[\begin{array}{ccc|cccc} & \mathbf{I}_{B_c} & & & \bar{\mathbf{M}} & & \\ \hline 0 & \dots & 0 & 1 & * & \dots & * \end{array}\right]$$

The last row would yield a loop in $\mathfrak{T}$ which leads to a contradiction. □

Lemma 3.37. *Let $\mathfrak{G}$ be an oriented graph with incidence matrix* $\mathbf{A}$ *and* $\mathfrak{T}$ *a forest in* $\mathfrak{G}$*. Let* $\mathfrak{s}_1, \dots, \mathfrak{s}_{B_t}$ *be the fundamental cutsets in* $\mathfrak{G}$ *that are defined by* $\mathfrak{T}$ *and let* $\mathbf{s}_1, \dots, \mathbf{s}_{B_t}$ *be the corresponding rows of the cutset matrix* $\mathbf{S}$*. If for an arbitrary cutset* $\mathfrak{s}$ *in* $\mathfrak{G}$ *the vector* $\mathbf{s}$ *is defined in analogy to* (3.7)*, then* $\mathbf{s}$ *is a linear combination of* $\mathbf{s}_1, \dots, \mathbf{s}_{B_t}$*.*

Proof. The lemma follows by an argument similar to the proof of Lemma 3.36. □

Remark 3.38. *Consider a graph $\mathfrak{G}$ which contains self loops. These self loops cannot be part of a tree in $\mathfrak{G}$. Since self loop only consist of the branch that closes the loop, Definitions 3.31 and 3.32 can be extended to include self loops. Moreover, self loops cannot be part of any cutset in $\mathfrak{G}$, hence the column in any cutset matrix which corresponds to a self loop will be zero. Thus, Lemmas 3.34, 3.36 and 3.37 still hold.*

3.2. Circuit equations

In the following, circuits with general nonlinear capacitances, resistances, inductances and voltage and current sources that satisfy the restrictions given in [25, 28] (cf. Appendix I), will be considered. For those circuits the charge and flux oriented modified nodal analysis as modeling method will be presented. For a more detailed introduction to circuit modeling methods see [23].

Consider a circuit. Its topology is completely described by its reduced incidence matrix $\mathbf{A}$. Let $\mathbf{i}$ be the vector of all branch currents, $\mathbf{v}$ the vector of all branch voltages and $\mathbf{e}$ the vector of node potentials in the circuit. Since we will use parts of the vector $\mathbf{i}$ as unknowns in our circuit model, we will denote the derivative of $\mathbf{x}$ with respect to time rather by $\frac{d}{dt}\mathbf{x}$ than by $\dot{\mathbf{x}}$ for the remainder of this chapter in order to keep the notation unambiguous.

The currents and voltages of a circuit are related by Kirchhoff's Current Law (KCL) and Kirchhoff's Voltage Law (KVL). The KCL states that the sum of all currents that enter a node is equal to zero, or for the whole circuit,

$$\mathbf{A}\mathbf{i} = \mathbf{0}. \tag{3.10}$$

The KVL on the other hand states that the sum of the voltages over branches that form a loop in the circuit is equal to zero. This translates into a relation between the branch voltages and the node potentials of the circuit

$$\mathbf{v} = \mathbf{A}^T\mathbf{e}. \tag{3.11}$$

We now split the circuit into its capacitive, resistive and inductive subgraphs and into those subgraphs that are defined by voltage and current sources. For these subgraphs, we define the vectors of branch currents $\mathbf{i}_*$, branch voltages $\mathbf{v}_*$ and the incidence matrix $\mathbf{A}_*$, $* \in \{C, R, L, V, I\}$ for the capacitive, resistive and inductive part and the parts that are defined by voltage and current sources. The respective terms for the whole circuit can be written as

$$\mathbf{i} = \left[\mathbf{i}_C^T, \mathbf{i}_R^T, \mathbf{i}_L^T, \mathbf{i}_V^T, \mathbf{i}_I^T\right]^T, \tag{3.12a}$$

$$\mathbf{v} = \left[\mathbf{v}_C^T, \mathbf{v}_R^T, \mathbf{v}_L^T, \mathbf{v}_V^T, \mathbf{v}_I^T\right]^T, \tag{3.12b}$$

$$\mathbf{A} = \left[\mathbf{A}_C, \mathbf{A}_R, \mathbf{A}_L, \mathbf{A}_V, \mathbf{A}_I\right]. \tag{3.12c}$$

Thus, (3.10) together with (3.12a) and (3.12c) yields

$$\mathbf{A}\mathbf{i} = \mathbf{A}_C\mathbf{i}_C + \mathbf{A}_R\mathbf{i}_R + \mathbf{A}_L\mathbf{i}_L + \mathbf{A}_V\mathbf{i}_V + \mathbf{A}_I\mathbf{i}_I = \mathbf{0}, \tag{3.13}$$

and (3.11) together with (3.12b) and (3.12c) yields

$$\mathbf{v} = \begin{bmatrix} \mathbf{v}_C \\ \mathbf{v}_R \\ \mathbf{v}_L \\ \mathbf{v}_V \\ \mathbf{v}_I \end{bmatrix} = \begin{bmatrix} \mathbf{A}_C^T\mathbf{e} \\ \mathbf{A}_R^T\mathbf{e} \\ \mathbf{A}_L^T\mathbf{e} \\ \mathbf{A}_V^T\mathbf{e} \\ \mathbf{A}_I^T\mathbf{e} \end{bmatrix}. \tag{3.14}$$

To derive (3.13) and (3.14), we have only used information about the topology of the circuit. Now, we try to describe the relations between the current through and the voltage across a branch by the behavior of the element that defines this branch. For the resistive branches of the circuit, i.e. the branches that are defined by resistances, we get the relation

$$\mathbf{i}_R = \mathbf{g}(\mathbf{v}_R, t), \tag{3.15}$$

The relations for the capacitive and the inductive branches, i.e. the branches that are defined by capacitances and inductances, are given by

$$\mathbf{i}_C = \frac{d}{dt}\mathbf{q},\ \mathbf{q} = \mathbf{q}_C(\mathbf{v}_C, t) \tag{3.16}$$

and

$$\mathbf{v}_L = \frac{d}{dt}\mathbf{\Phi}.\ \mathbf{\Phi} = \mathbf{\Phi}_L(\mathbf{i}_L, t) \tag{3.17}$$

Here, $\mathbf{q}$ is the vector of charges of the capacitances and $\mathbf{\Phi}$ is the vector of magnetic fluxes in the inductances of the circuit. For further use, we will define the partial derivatives

$$\begin{aligned} \mathbf{C}(\mathbf{v}_C, t) &:= \frac{\partial}{\partial \mathbf{v}_C}\mathbf{q}_C(\mathbf{v}_C, t), & \mathbf{L}(\mathbf{i}_L, t) &:= \frac{\partial}{\partial \mathbf{i}_L}\mathbf{\Phi}_L(\mathbf{i}_L, t), & \mathbf{G}(\mathbf{v}_R, t) &:= \frac{\partial}{\partial \mathbf{v}_R}\mathbf{g}(\mathbf{v}_R, t), \\ \mathbf{q}'_t(\mathbf{v}_C, t) &:= \frac{\partial}{\partial t}\mathbf{q}_C(\mathbf{v}_C, t), & \mathbf{\Phi}'_t(\mathbf{i}_L, t) &:= \frac{\partial}{\partial t}\mathbf{\Phi}_L(\mathbf{i}_L, t), & \mathbf{g}'_t(\mathbf{v}_R, t) &:= \frac{\partial}{\partial t}\mathbf{g}(\mathbf{v}_R, t). \end{aligned} \tag{3.18}$$

The matrix $\mathbf{C}(\mathbf{v}_C, t)$ is called the *capacitance matrix*, $\mathbf{L}(\mathbf{i}_L, t)$ is the *inductance* and $\mathbf{G}(\mathbf{v}, t)$ the *conductance matrix* of the considered circuit. Finally, the current and voltage sources are given by

$$\mathbf{i}_I = \mathbf{i}_s\left(\mathbf{A}^T\mathbf{e}, \frac{d}{dt}\mathbf{q}_C\left(\mathbf{A}_C^T\mathbf{e}, t\right), \mathbf{i}_L, \mathbf{i}_V, t\right) =: \mathbf{i}_s(*, t) \tag{3.19}$$

and

$$\mathbf{v}_V = \mathbf{v}_s\left(\mathbf{A}^T\mathbf{e}, \frac{d}{dt}\mathbf{q}_C\left(\mathbf{A}_C^T\mathbf{e}, t\right), \mathbf{i}_L, \mathbf{i}_V, t\right) =: \mathbf{v}_s(*, t) \tag{3.20}$$

(cf. [27]). For the remained of this thesis, we will assume that Assumption A1 holds.

Assumption A1. *For the circuit under consideration it holds that*

1. *the controlled sources in the circuit fulfill the conditions which are displayed in Appendix I.*
2. *the matrices* $\mathbf{C}(\mathbf{v}, t)$, $\mathbf{G}(\mathbf{v}, t)$ *and* $\mathbf{L}(\mathbf{i}, t)$ *are positive definite for all* $\mathbf{v}$, $\mathbf{i}$ *and* t.

Combining the branch relations (3.15), (3.16) and (3.17) and the description of the sources (3.19) and (3.20) with the information about the topology of the circuit, which is given by (3.13) and (3.14), yields the following system

$$\mathbf{0} = \mathbf{A}_C\frac{d}{dt}\mathbf{q} + \mathbf{A}_R\mathbf{g}\left(\mathbf{A}_R^T\mathbf{e}, t\right) + \mathbf{A}_L\mathbf{i}_L + \mathbf{A}_V\mathbf{i}_V + \mathbf{A}_I\mathbf{i}_s(*, t), \tag{3.21a}$$

$$\mathbf{0} = \frac{d}{dt}\mathbf{\Phi} - \mathbf{A}_L^T\mathbf{e}, \tag{3.21b}$$

$$\mathbf{0} = \mathbf{A}_V^T\mathbf{e} - \mathbf{v}_s(*, t), \tag{3.21c}$$

$$\mathbf{0} = \mathbf{q} - \mathbf{q}_C\left(\mathbf{A}_C^T\mathbf{e}, t\right), \tag{3.21d}$$

$$\mathbf{0} = \mathbf{\Phi} - \mathbf{\Phi}_L(\mathbf{i}_L, t), \tag{3.21e}$$

where $\mathbf{q}$, $\mathbf{\Phi}$, $\mathbf{e}$, $\mathbf{i}_L$ and $\mathbf{i}_V$ are the unknowns of the system. The procedure outlined above is the charge and flux oriented modified nodal analysis (MNA c/f). It yields a mixed system of differential and algebraic equations (DAE). If the equations (3.21d) and (3.21e) are used to eliminate the charges $\mathbf{q}$ and the fluxes $\mathbf{\Phi}$ from system (3.21), then this yields

$$\mathbf{0} = \mathbf{A}_C \frac{d}{dt}\mathbf{q}_C\left(\mathbf{A}_C^T\mathbf{e}, t\right) + \mathbf{A}_R\mathbf{g}\left(\mathbf{A}_R^T\mathbf{e}, t\right) + \mathbf{A}_L\mathbf{i}_L + \mathbf{A}_V\mathbf{i}_V + \mathbf{A}_I\mathbf{i}_s(*, t), \tag{3.22a}$$

$$\mathbf{0} = \frac{d}{dt}\mathbf{\Phi}_L(\mathbf{i}_L, t) - \mathbf{A}_L^T\mathbf{e}, \tag{3.22b}$$

$$\mathbf{0} = \mathbf{A}_V^T\mathbf{e} - \mathbf{v}_s(*, t) \tag{3.22c}$$

with the unknowns $\mathbf{e}$, $\mathbf{i}_L$ and $\mathbf{i}_V$. System (3.22) represents the circuit in the case that it is modeled by the standard modified nodal analysis (MNA). It is usually much smaller due to the fact that the conservation laws for the capacitive charges $\mathbf{q}$ and the inductive fluxes $\mathbf{\Phi}$ have been omitted. However, this also may lead to numerical instabilities during the integration due to the possible non-conservation of charges and fluxes.

Theorem 3.39. *Both* (3.21) *and* (3.22) *have properly stated leading terms.*

Proof. Cf. [59]. □

3.3. The index of circuit equations

Due to the special structure of electrical circuit equations, it is possible to determine the differentiation index and those parts of the system that lead to the so called hidden constraints by graph theoretical considerations. In order to be able to apply results from graph theory, we identify each element of a circuit with a branch in a graph. The resulting graph is called the *network graph* of the circuit. The first part of this section gives the results of [28]. In the second part, these results will be examined more closely from a graph theoretical point of view.

3.3.1. Index determination

We start by briefly summarizing the results of [28]. To this end, the following definitions are given.

Definition 3.40. (LI cutset, CV loop) Consider a circuit graph.

1. A cutset which consists of branches of inductances and which may also contain branches of current sources is called *LI cutset.*

2. A loop which consists of branches of capacitances and branches of voltage sources and which contains at least one voltage source is called *CV loop.*

□

To identify certain loops and cutsets of the circuit we define the following matrices (cf. [28,48]).

Definition 3.41. Consider either the system (3.21) or the system (3.22) with component related incidence matrices $\mathbf{A}_C$, $\mathbf{A}_R$, $\mathbf{A}_V$ and $\mathbf{A}_L$. Then, $\mathbf{Z}_C$, $\mathbf{Z}_{V-C}$ and $\mathbf{Z}_{R-CV}$ are bases of $\ker \mathbf{A}_C^T$, $\ker \mathbf{A}_V^T \mathbf{Z}_C$ and $\ker \mathbf{A}_R^T \mathbf{Z}_C \mathbf{Z}_{V-C}$, respectively. The matrix $\bar{\mathbf{Z}}_{V-C}$ is a basis of $\ker \mathbf{Z}_C^T \mathbf{A}_V$ and the product $\mathbf{Z}_C \mathbf{Z}_{V-C} \mathbf{Z}_{R-CV}$ is denoted by $\mathbf{Z}_{CRV}$.

Here, a matrix is said to be a basis of a subspace, if this is true for the columns of the matrix (cf. [50]).

Remark 3.42. *Note, that the matrices defined in e.g. [25, 28] are projectors onto the subspaces, instead of bases of these subspaces.*

The circuit is assumed to contain neither loops that consist only of voltage sources nor cutsets that consist only of current sources, because both configurations may contradict Kirchhoff's laws. Furthermore, controlled sources are not allowed to be part of CV loops or LI cutsets. The controlling elements of both kinds of controlled sources have to fulfill the conditions given in Appendix I. Under these assumptions, it is possible to show that the differentiation index of the DAE (3.21) or (3.22) is always less than or equal to 2. The following theorem of [28] now provides information about the relation between the network graph of a circuit and the differentiation index of the DAE arising from either MNA or MNA c/f and shows which part of the systems (3.21) or (3.22) have to be differentiated to obtain the hidden constraints.

Theorem 3.43. *(cf. [28]) Consider a DAE that arises either from MNA or from MNA c/f and assume that Assumption A1 holds.*

1. *If the circuit does neither contain CV loops nor LI cutsets, then the differentiation index of the DAE is equal to* 1 *and the algebraic constraints are given by*

$$\begin{aligned}\mathbf{0} &= \mathbf{Z}_C^T \left[\mathbf{A}_R \mathbf{g}\left(\mathbf{A}_R^T \mathbf{e}, t\right) + \mathbf{A}_L \mathbf{i}_L + \mathbf{A}_V \mathbf{i}_V + \mathbf{A}_I \mathbf{i}_s(*, t)\right],\\ \mathbf{0} &= \mathbf{A}_V^T \mathbf{e} - \mathbf{v}_s(*, t),\end{aligned}$$

   in case of MNA and

$$\begin{aligned}\mathbf{0} &= \mathbf{Z}_C^T \left[\mathbf{A}_R \mathbf{g}\left(\mathbf{A}_R^T \mathbf{e}, t\right) + \mathbf{A}_L \mathbf{i}_L + \mathbf{A}_V \mathbf{i}_V + \mathbf{A}_I \mathbf{i}_s(*, t)\right],\\ \mathbf{0} &= \mathbf{A}_V^T \mathbf{e} - \mathbf{v}_s(*, t),\\ \mathbf{0} &= \mathbf{q} - \mathbf{q}_C\left(\mathbf{A}_C^T \mathbf{e}, t\right),\\ \mathbf{0} &= \mathbf{\Phi} - \mathbf{\Phi}_L(\mathbf{i}_L, t),\end{aligned}$$

   in case of MNA c/f.

2. *If the circuit contains CV loops, but no LI cutsets, then the differential index is equal to* 2, $\bar{\mathbf{Z}}_{V-C}$ *is a non-zero matrix and the derivatives that are needed to determine the hidden constraints are*

$$\mathbf{0} = \bar{\mathbf{Z}}_{V-C}^T \left(\mathbf{A}_V^T \frac{d}{dt}\mathbf{e} - \frac{d}{dt}\mathbf{v}_s(*, t)\right) \qquad (3.23)$$

in case of MNA and

$$\mathbf{0} = \bar{\mathbf{Z}}_{V-C}^T \left(\mathbf{A}_V^T \frac{d}{dt}\mathbf{e} - \frac{d}{dt}\mathbf{v}_s(*,t) \right), \tag{3.24a}$$

$$\mathbf{0} = \frac{d}{dt}\mathbf{q} - \frac{d}{dt}\mathbf{q}_C\left(\mathbf{A}_C^T\mathbf{e}, t\right) \tag{3.24b}$$

in case of MNA c/f.

3. *If the circuit contains LI cutsets, but no CV loops, then the differentiation index is equal to* 2, $\mathbf{Z}_{CRV}$ *is a non-zero matrix and the derivatives that are needed to determine the hidden constraints are*

$$\mathbf{0} = \mathbf{Z}_{CRV}^T \left(\mathbf{A}_L \frac{d}{dt}\mathbf{i}_L + \mathbf{A}_I \frac{d}{dt}\mathbf{i}_s(*,t) \right) \tag{3.25}$$

in case of MNA and

$$\mathbf{0} = \mathbf{Z}_{CRV}^T \left(\mathbf{A}_L \frac{d}{dt}\mathbf{i}_L + \mathbf{A}_I \frac{d}{dt}\mathbf{i}_s(*,t) \right), \tag{3.26a}$$

$$\mathbf{0} = \frac{d}{dt}\mathbf{\Phi} - \frac{d}{dt}\mathbf{\Phi}_L(\mathbf{i}_L, t) \tag{3.26b}$$

in case of MNA c/f.

4. *If the circuit contains both CV loops and LI cutsets, then both matrices* $\bar{\mathbf{Z}}_{V-C}$ *and* $\mathbf{Z}_{CRV}$ *are non-zero matrices and the differentiation index is equal to* 2, *independent of the formulation. To derive the hidden constraints,* (3.23) *and* (3.25) *are needed in case of MNA and* (3.24) *and* (3.26) *in case of MNA c/f.*

Remark 3.44. *If the DAE* (3.22) *has differentiation index 2, then we see from equations* (3.23) *and* (3.25) *that in order to be able to compute the hidden constraints the functions of the sources that belong to CV loops or LI cutsets have to be differentiable. Moreover, if we consider a DAE of the form* (3.21) *with differentiation index 2, then also the functions* $\mathbf{q}_C(\mathbf{v}_C, t)$ *and* $\mathbf{\Phi}_L(\mathbf{i}_L, t)$, *that describe the capacitive charges and the inductive fluxes, have to be differentiable.*

3.3.2. The matrices $\bar{\mathbf{Z}}_{C-V}$ and $\mathbf{Z}_{CRV}$

The matrices $\bar{\mathbf{Z}}_{V-C}$ and $\mathbf{Z}_{CRV}$ will play an important role in the following sections, so it is advantageous to express them in terms of matrices that are related to graphs. To be able to find such a characterization, we will take a closer look on the actions of the matrices $\mathbf{Z}_C$, $\mathbf{Z}_{V-C}$ and $\mathbf{Z}_{R-CV}$ on the network graph $\mathfrak{G}$ of the circuit under consideration.

The matrix $\mathbf{Z}_C$

The matrix $\mathbf{Z}_C$ is defined to be a basis of $\ker \mathbf{A}_C^T$. Consider the subgraph of $\mathfrak{G}$ that contains all nodes and all capacitive branches of $\mathfrak{G}$. We will call this subgraph the *C-subgraph* of $\mathfrak{G}$ and denote it by $\mathfrak{G}_C$. Then, $\mathbf{A}_C$ is the incidence matrix of $\mathfrak{G}_C$ and we can define $\mathbf{Z}_C$ by applying Lemma 3.29 to $\mathfrak{G}_C$.

The matrices $\mathbf{Z}_{V-C}$ and $\mathbf{Z}_C\mathbf{Z}_{V-C}$

The matrix $\mathbf{Z}_{V-C}$ is defined to be a basis of $\ker \mathbf{A}_V^T\mathbf{Z}_C$, hence we need to find a graph theoretical interpretation of $\mathbf{A}_V^T\mathbf{Z}_C$. In order to find such an interpretation we consider again $\mathbf{Z}_C$. With Remark 3.30, we are able to interpret $\mathbf{Z}_C^T\mathbf{A}$ as the incidence matrix of the graph $\mathfrak{G}_{-C}$ that results from $\mathfrak{G}$ if the $\mathfrak{G}_C$ is contracted. If $\mathfrak{G}_{V-C}$ is the part of $\mathfrak{G}_{-C}$ that contains all nodes of $\mathfrak{G}_{-C}$ and all branches that are defined by voltage sources, then $\mathbf{Z}_C^T\mathbf{A}_V$ is the part of the incidence matrix $\mathbf{Z}_C^T\mathbf{A}$ that belongs to $\mathfrak{G}_{V-C}$ and $\mathbf{Z}_{V-C}$ can be determined by applying Lemma 3.29 to $\mathfrak{G}_{V-C}$.

Moreover, it is possible to determine the product $\mathbf{Z}_{CV} := \mathbf{Z}_C\mathbf{Z}_{V-C}$ directly. To see this, we consider the graph $\mathfrak{G}_{CV}$ which is the subgraph of $\mathfrak{G}$ that contains all nodes and all branches that are either defined by capacitances or by voltage sources. Multiplying $\mathbf{A}$ from the left first by $\mathbf{Z}_C^T$ and then by $\mathbf{Z}_{V-C}^T$ is then equivalent to contracting the subgraph $\mathfrak{G}_{CV}$. Hence, we obtain $\mathbf{Z}_{CV}^T$ if we apply Lemma 3.29 to $\mathfrak{G}_{CV}$.

Example 3.45. To illustrate the effect of the contraction of the C-subgraph, we consider the circuit that is shown in Figure 3.3.

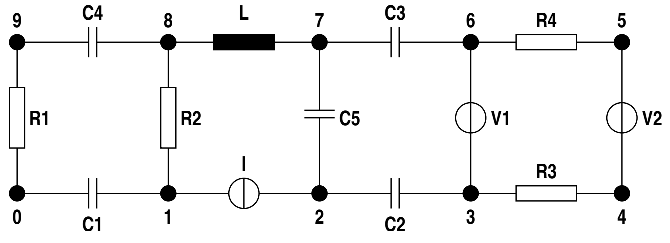


Figure 3.3.: Example circuit

In Figure 3.3, the capacitances C_2, C_3 and C_5 form a component of the C-subgraph $\mathfrak{G}_C$. Two more components consist of the capacitances C_1 and C_4. In addition, the nodes 4 and 5 have to be considered as components of $\mathfrak{G}_C$. Since C_1 is incident with the reference node, the corresponding component will not be taken into account for the determination of $\mathbf{Z}_C$. Therefore, $\mathbf{Z}_C$ is given by

$$\mathbf{Z}_C = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}^T .$$

To determine $\mathbf{Z}_{V-C}$, we apply the contraction to the network graph in Figure 3.3 and obtain the network graph shown in Figure 3.4 where capacitive self loops have been omitted for clarity reasons.

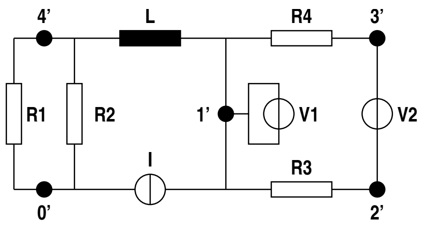


Figure 3.4.: Example circuit after contraction of the C-subgraph

The subgraph $\mathfrak{G}_{V-C}$ consists of the remaining voltage source V_2 and of the nodes $0'$, $1'$ and $4'$. Again, the node $0'$ is omitted and we obtain $\mathbf{Z}_{V-C}$ as

$$\mathbf{Z}_{V-C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^T .$$

Hence, the product $\mathbf{Z}_C\mathbf{Z}_{V-C}$ is given by

$$\mathbf{Z}_C\mathbf{Z}_{V-C} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}^T .$$

If we consider the subgraph $\mathfrak{G}_{CV}$ in Figure 3.3, then we obtain the same matrix $\mathbf{Z}_{CV}$. □

The matrices $\mathbf{Z}_{R-CV}$ and $\mathbf{Z}_{CRV}$

Let $\mathfrak{G}_{R-CV}$ be the subgraph of $\mathfrak{G}_{-CV}$ that consists of all nodes of $\mathfrak{G}_{-CV}$ and all resistive branches. The matrix $\mathbf{Z}_{R-CV}$ can be derived from $\mathfrak{G}_{R-CV}$ in the same way as $\mathbf{Z}_{V-C}$ is derived from $\mathfrak{G}_{V-C}$. Alternatively, we can determine the product $\mathbf{Z}_{CRV} = \mathbf{Z}_C\mathbf{Z}_{V-C}\mathbf{Z}_{R-CV}$ by applying Lemma 3.29 to the subgraph $\mathfrak{G}_{CRV}$ of $\mathfrak{G}$ that contains all nodes and all branches that are either defined by resistances, capacitances or voltage sources. As before, the multiplication of $\mathbf{A}$ with $\mathbf{Z}_{CRV}^T$ from the left is then equivalent to contracting the subgraph $\mathfrak{G}_{CRV}$ and $\mathbf{Z}_{CRV}^T\mathbf{A}$ is the incidence matrix of the resulting graph $\mathfrak{G}_{-CRV}$. The branches in graph $\mathfrak{G}_{-CRV}$ that are not self loops have to be inductive branches and branches that are defined by current sources. The nodes of $\mathfrak{G}_{-CRV}$ are the components of $\mathfrak{G}_{CRV}$. The reference node of $\mathfrak{G}_{-CRV}$ corresponds to the component of $\mathfrak{G}_{CRV}$ that contains the original reference node.

We will now examine $\mathbf{Z}_{CRV}^T\,[\mathbf{A}_I\ \mathbf{A}_L]$ closer. This is the non-trivial part of the incidence matrix of $\mathfrak{G}_{-CRV}$. Each row contains information about which branches enter or leave the respective node. On the other hand, all branches that enter or leave a node belong to the cutset that separates the node from the remaining graph. If $\mathfrak{G}_{-CRV}$ contains N_{-CRV}

nodes, then $\mathbf{Z}_{CRV}^T [\mathbf{A}_L \ \mathbf{A}_I]$ yields $N_{LI} := N_{-CRV} - 1$ such cutsets. Since $\mathbf{Z}_{CRV}^T [\mathbf{A}_L \ \mathbf{A}_I]$ has row rank N_{LI}, the rows are linearly independent and we are able to choose $\mathbf{Z}_{CRV}$ such that

$$\mathbf{Z}_{CRV}^T [\mathbf{A}_I \ \mathbf{A}_L] = \mathbf{S}_{LI}, \tag{3.27}$$

where $\mathbf{S}_{LI}$ is a fundamental cutset matrix of $\mathfrak{G}_{-CRV}$. Every cutset in $\mathfrak{G}_{-CRV}$ is an LI cutset in $\mathfrak{G}$, which clarifies the connection between $\mathbf{Z}_{CRV}$ and the LI cutsets in $\mathfrak{G}$ that lead to hidden constraints.

To further examine the structure of $\mathbf{S}_{LI}$, we first note that cutsets that consist of current sources exclusively are not allowed. We also know from the conditions in Appendix I that only independent sources are allowed in LI cutsets. Hence we can always find a tree of inductive branches in $\mathfrak{G}_{-CRV}$ and the reordering from Lemma 3.34 can be done in such a way, that

$$\mathbf{S}_{LI} = \Big[\underbrace{\mathbf{S}_{I,ind}}_{\substack{\text{independent}\\ \text{current}\\ \text{sources}}} \quad \underbrace{\mathbf{0}_{I,contr}}_{\substack{\text{controlled}\\ \text{current}\\ \text{sources}}} \quad \underbrace{\widetilde{\mathbf{S}}_L}_{\substack{\text{inductive}\\ \text{connecting}\\ \text{branches}}} \quad \underbrace{\mathbf{0}_L}_{\substack{\text{inductive}\\ \text{branches}\\ \text{outside}\\ \text{LI cut-}\\ \text{sets}}} \quad \underbrace{\mathbf{I}_{N_{LI}}}_{\substack{\text{inductive}\\ \text{tree}\\ \text{branches}}} \Big]. \tag{3.28}$$

This special reordering induces a splitting of the incidence matrices $\mathbf{A}_I$ and $\mathbf{A}_L$. We will assume without loss of generality that the respective branches are already ordered such that $\mathbf{A}_I$ and $\mathbf{A}_L$ can be written as

$$\mathbf{A}_I = [\mathbf{A}_{I,ind} \ \mathbf{A}_{I,contr}], \tag{3.29a}$$

$$\mathbf{A}_L = \left[\widetilde{\mathbf{A}}_L \ \bar{\mathbf{A}}_L \ \widehat{\mathbf{A}}_L\right], \tag{3.29b}$$

such that the parts of the incidence matrices are defined by the sets of branches that define the splitting in (3.28). Now, (3.27) together with (3.29) yields the following identities which will be crucial for the following index reduction method.

$$\mathbf{S}_{I,ind} = \mathbf{Z}_{CRV}^T \mathbf{A}_{I,ind}, \quad \mathbf{0}_{I,contr} = \mathbf{Z}_{CRV}^T \mathbf{A}_{I,contr}, \tag{3.30a}$$

$$\widetilde{\mathbf{S}}_L = \mathbf{Z}_{CRV}^T \widetilde{\mathbf{A}}_L, \quad \mathbf{0}_L = \mathbf{Z}_{CRV}^T \bar{\mathbf{A}}_L, \quad \mathbf{I}_{N_{LI}} = \mathbf{Z}_{CRV}^T \widehat{\mathbf{A}}_L. \tag{3.30b}$$

The matrix $\bar{\mathbf{Z}}_{V-C}$

The matrix $\bar{\mathbf{Z}}_{V-C}$ is rather easy to characterize. We have already stated that $\mathbf{Z}_C^T \mathbf{A}_V$ is the incidence matrix of the subgraph $\mathfrak{G}_{V-C}$. Since $\bar{\mathbf{Z}}_{V-C}$ should span the kernel of $\mathbf{Z}_C^T \mathbf{A}_V$, we can use Corollary 3.27 and choose $\bar{\mathbf{Z}}_{V-C} \in \mathbb{R}^{N_V \times N_{CV}}$ to be the (fundamental) loop matrix of $\mathfrak{G}_{V-C}$. Here, N_V is the number of voltage sources in the circuit and N_{CV} is the number of fundamental loops in $\mathfrak{G}_{V-C}$. We know that $\mathfrak{G}$ does not contain any loops that only consist of voltage sources (V loop) and that the nodes of $\mathfrak{G}_{-C}$ are the components of $\mathfrak{G}_C$. Therefore, any V loop in $\mathfrak{G}_{V-C}$ is a CV loop in $\mathfrak{G}$, which again shows the connection between $\bar{\mathbf{Z}}_{V-C}$ and the CV loops in $\mathfrak{G}$ that yield hidden constraints.

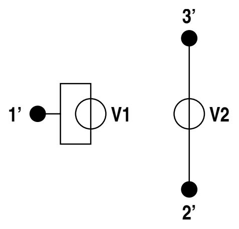


Figure 3.5.: The subgraph $\mathfrak{G}_{V-C}$

Example 3.46. We consider again the network graph displayed in Figure 3.3. After the contraction on the C-subgraphs, we get the network graph in Figure 3.4. The subgraph $\mathfrak{G}_{V-C}$ is shown in Figure 3.5.

The graph $\mathfrak{G}_{V-C}$ contains only one V loop which consists of the voltage source V_1. Comparing Figure 3.5 with the network graph in Figure 3.3, we see that V_1 forms a CV loop with the capacitances C_2, C_3 and C_5. The matrix $\bar{\mathbf{Z}}_{V-C}$ for the network graph in this example is given by

$$\bar{\mathbf{Z}}_{V-C}^T = \begin{bmatrix} 1 & 0 \end{bmatrix}.$$

□

4. Index reduction in circuit simulation

In Section 3.3 it has been shown that the differentiation index of a circuit DAE arising from the Modified Nodal Analysis or the charge-oriented Modified Nodal Analysis is tied to the network graph of the circuit. Also, the derivatives that lead to higher index can be determined based on the network graph. In this chapter this information will be used to reduce the differentiation index of circuit DAEs. The discussion starts by examining the influence of controlled sources in Section 4.1. Section 4.2 then presents an index reduction method suitable for circuit DAEs from MNA. This index reduction method uses the derivatives as given in equations (3.23) and (3.25). However, an analogous approach for circuit DAEs from MNA c/f is not directly possible. The method presented in Section 4.3 therefore tries to determine the necessary derivatives in such a way that these derivatives are equivalent to equations (3.24) and (3.26b), but more suited for an index reduction of a DAE from MNA c/f. Section 4.4 shows that the index reduction method for DAEs from MNA c/f as proposed in Section 4.3 can be interpreted as direct modifications to a given circuit that leads to a DAE with differentiation index 1.

4.1. Controlled sources and index reduction

In Section 4.3 and Section 4.4 two index reduction methods will be proposed that rely on direct changes to the respective circuit. If such a change affects the controlling element of a controlled source, we have to make sure that the controlled source still fulfills the condition in Appendix I.

Theorem 4.1. *Let $\mathfrak{G}$ be a network graph in which the controlled sources fulfill the conditions in Appendices I.1 and I.2. Moreover, let $\mathfrak{G}'$ be the network graph that results from $\mathfrak{G}$ by inserting an independent voltage source in series with each capacitance that belongs to a CV loop. Then, the controlled sources in $\mathfrak{G}'$ also fulfill the conditions in Appendices I.1 and I.2.*

Proof. See Appendix I.3. □

Theorem 4.2. *Let $\mathfrak{G}$ be a network graph in which the controlled sources fulfill the conditions in Appendices I.1 and I.2. Moreover, let $\mathfrak{G}'$ be the network graph that results from $\mathfrak{G}$ if one capacitance in each CV loop is replaced by a controlled current source. Then, the controlled sources in $\mathfrak{G}'$ that correspond to the controlled sources in $\mathfrak{G}$ also fulfill the conditions in Appendices I.1 and I.2.*

Proof. See Appendix I.4. □

Theorem 4.3. *Let $\mathfrak{G}$ be a network graph in which the controlled sources fulfill the conditions in Appendices I.1 and I.2. Moreover, let $\mathfrak{G}'$ be the network graph that results from $\mathfrak{G}$ if one inductance in each LI cutset is replaced by a controlled voltage source. Then, the controlled sources in $\mathfrak{G}'$ that correspond to the controlled sources in $\mathfrak{G}$ also fulfill the conditions in Appendices I.1 and I.2.*

Proof. See Appendix I.5. □

4.2. Index reduction for MNA equations

In this section, we will develop algebraic index reduction methods for circuits that have been modeled with MNA. We will present methods which are based on the index reduction methods by minimal extention which have been proposed in [48]. This method will be adapted to the special structure of the DAEs that arise from either of the modeling techniques. To simplify the notation, we will examine the cases of CV loops and LI cutsets separately. Note that it is possible to combine the index reduction methods, if a circuit contains both CV loops and LI cutsets.

4.2.1. Index reduction for circuits with CV loops

We will consider a circuit with controlled sources that fulfill the conditions listed in Appendices I.1 and I.2 and assume that it only contains CV loops, but no LI cutsets. Hence, due to Theorem 3.43, equation (3.23) yields hidden constraints and has to be added to the DAE (3.22). In this way, we obtain the extended system

$$\mathbf{0} = \mathbf{A}_C \frac{d}{dt}\mathbf{q}_C\left(\mathbf{A}_C^T\mathbf{e}, t\right) + \mathbf{A}_R\mathbf{g}\left(\mathbf{A}_R^T\mathbf{e}, t\right) + \mathbf{A}_L\mathbf{i}_L + \mathbf{A}_V\mathbf{i}_V + \mathbf{A}_I\mathbf{i}_s(*, t), \tag{4.1a}$$

$$\mathbf{0} = \frac{d}{dt}\mathbf{\Phi}_L(\mathbf{i}_L, t) - \mathbf{A}_L^T\mathbf{e}, \tag{4.1b}$$

$$\mathbf{0} = \mathbf{A}_V^T\mathbf{e} - \mathbf{v}_s(*, t), \tag{4.1c}$$

$$\mathbf{0} = \bar{\mathbf{Z}}_{V-C}^T\mathbf{A}_{V,ind}^T\frac{d}{dt}\mathbf{e} - \bar{\mathbf{Z}}_{V-C}^T\mathbf{v}_{ind}(t). \tag{4.1d}$$

Here, $\mathbf{A}_{V,ind}$ denotes the incidence matrix with respect to independent voltage sources and $\mathbf{v}_{ind}(t)$ denotes the functions that describe the independent voltage sources. The restriction to independent voltage sources in equation (4.1d) is possible since controlled voltage sources are not allowed to be part of CV loops (cf. Appendix I.1). Our first objective will be to determine the derivatives (4.1d) in such a way that these equations can be easily inserted into the original DAE system.

Determination of required derivatives

Both Algorithm 1 and Algorithm 2, which will be presented in the following, are taken from [24]. In the following, $\mathfrak{G}_C$ denotes again the capacitive subgraph of $\mathfrak{G}$. Note that, in contrast to Algorithm 1 and Section 3.3, the connected components of $\mathfrak{G}_C$ in Algorithm 2 do not include isolated nodes, i.e. nodes that are not incident with any branch.

Algorithm 1: Computation of $\bar{\mathbf{Z}}_{V-C}^T$ – Part 1

```
  Data: Network graph $\mathfrak{G}$
  begin
1     Set $\widetilde{\mathfrak{G}} = \mathfrak{G}$;
2     while a CV loop $\mathfrak{m}$ is found in $\widetilde{\mathfrak{G}}$ do
3         save the voltage sources and the components of $\mathfrak{G}_C$ that are part of $\mathfrak{m}$;
4         set $\widetilde{\mathfrak{G}}$ to the graph obtained by deleting one of the voltage sources in $\mathfrak{m}$ and
          contracting the incident nodes;
5     define the coefficients of $\bar{\mathbf{Z}}_{V-C}^T$ by (3.6) restricted to the sources found in step 3;
6     color all nodes in $\mathfrak{G}$ that are not incident with capacitances;
7     foreach connected component $\mathfrak{C}$ of $\mathfrak{G}_C$ do
8         if $\mathfrak{C}$ does not contain the datum node then color an arbitrary node;
9         else color the datum node;
  end
```

With Algorithm 1 the hidden constraints are given by equation (3.23)

$$\mathbf{0} = \bar{\mathbf{Z}}_{V-C}^T \left(\mathbf{A}_V^T \frac{d}{dt}\mathbf{e} - \frac{d}{dt}\mathbf{v}_s(*,t) \right).$$

Moreover, if $\widehat{\mathbf{e}}$ is the vector of node potentials of those nodes colored in step 7 of Algorithm 2 and $\widehat{\mathbf{A}}_V$ the part of $\mathbf{A}_V$ related to those nodes, then $\bar{\mathbf{Z}}_{V-C}^T \widehat{\mathbf{A}}_V^T$ is nonsingular, cf. proof of Theorem 7.6 in [24].

Algebraic transformation of the circuit DAE

To determine those differential variables that have to be replaced by algebraic variables we need to find permutations $\mathbf{\Pi}_e$ such that

$$\bar{\mathbf{Z}}_{V-C}^T \mathbf{A}_{V,ind}^T \mathbf{\Pi}_e^T = [\mathbf{F}_1 \ \mathbf{F}_2], \tag{4.2}$$

with $\mathbf{F}_1$ nonsingular. As shown in the previous section, Algorithm 2 can be used to obtain such a permutation. We permute the node potentials $\mathbf{e}$ accordingly and split $\mathbf{e}$ into

$$\mathbf{\Pi}_e \mathbf{e} =: \widetilde{\mathbf{e}} =: \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \end{bmatrix}.$$

Following the idea of the index reduction by minimal extension method [48], we see that we have to set

$$\frac{d}{dt}\mathbf{e}_1 =: \widehat{\mathbf{e}}_1. \tag{4.3}$$

We now multiply (4.1a) from the left by $\mathbf{\Pi}_e$ and and apply (4.2) and (4.3) to (3.22). To simplify the notation we set $\widetilde{\mathbf{A}}_* := \mathbf{\Pi}_e \mathbf{A}_*$, $* \in \{C, R, L, V, I\}$ and $\widetilde{\mathbf{A}} := \mathbf{\Pi}_e \mathbf{A}$. In this

Algorithm 2: Computation of $\bar{\mathbf{Z}}_{V-C}^T$ – Part 2

```
   Data: Network graph $\mathfrak{G}$
   begin
1      Set $\widetilde{\mathfrak{G}}$ to the subgraph of $\mathfrak{G}$ that consists of the voltage sources found in step 3
       Algorithm 1;
2      foreach connected component $\mathfrak{C}$ of $\mathfrak{G}_C$ do
3          while $\mathfrak{C}$ still forms unexamined CV loops with $\widetilde{\mathfrak{G}}$ do
4              if an colored node of $\mathfrak{C}$ is incident with $\widetilde{\mathfrak{G}}$ and part of a CV loop $\mathfrak{m}$ then
               make $\mathfrak{m}$ the current CV loop;
5              else choose an arbitrary CV loop $\mathfrak{m}$ as current loop;
6              choose an uncolored node in which $\mathfrak{m}$ is incident with branches of $\widetilde{\mathfrak{G}}$;
7              color that node;
8              delete from $\widetilde{\mathfrak{G}}$ the voltage source in $\mathfrak{m}$ that is incident with node colored
               in step 7;
9              mark the current loop as examined;
10         add $\mathfrak{C}$ to $\widetilde{\mathfrak{G}}$;
   end
```

way, we obtain the system

$$\begin{aligned}
\mathbf{0} &= \widetilde{\mathbf{A}}_C\mathbf{C}\left(\widetilde{\mathbf{A}}_C^T\widetilde{\mathbf{e}},t\right)\widetilde{\mathbf{A}}_C^T\begin{bmatrix}\widehat{\mathbf{e}}_1\\ \frac{d}{dt}\mathbf{e}_2\end{bmatrix} + \widetilde{\mathbf{A}}_C\mathbf{q}_t'\left(\widetilde{\mathbf{A}}_C^T\widetilde{\mathbf{e}},t\right)\\
&\quad + \widetilde{\mathbf{A}}_R\mathbf{g}\left(\widetilde{\mathbf{A}}_R^T\widetilde{\mathbf{e}},t\right) + \widetilde{\mathbf{A}}_L\mathbf{i}_L + \widetilde{\mathbf{A}}_V\mathbf{i}_V + \widetilde{\mathbf{A}}_I\mathbf{i}_s(*,t), && (4.4a)\\
\mathbf{0} &= \frac{d}{dt}\mathbf{\Phi}_L(\mathbf{i}_L,t) - \widetilde{\mathbf{A}}_L^T\widetilde{\mathbf{e}}, && (4.4b)\\
\mathbf{0} &= \widetilde{\mathbf{A}}_V^T\widetilde{\mathbf{e}} - \mathbf{v}_s(*,t), && (4.4c)\\
\mathbf{0} &= \mathbf{F}_2\frac{d}{dt}\mathbf{e}_2 + \mathbf{F}_1\widehat{\mathbf{e}}_1 - \bar{\mathbf{Z}}_{V-C}^T\frac{d}{dt}\mathbf{v}_{ind}(t), && (4.4d)
\end{aligned}$$

which according to Section 2.4 has differentiation index 1. Since $\mathbf{F}_1$ is nonsingular, we can solve (4.4d) for $\widehat{\mathbf{e}}_1$ and obtain

$$\widehat{\mathbf{e}}_1 = \mathbf{F}_1^{-1}\left(\bar{\mathbf{Z}}_{V-C}^T\frac{d}{dt}\mathbf{v}_{ind}(t) - \mathbf{F}_2\frac{d}{dt}\mathbf{e}_2\right). \tag{4.5}$$

We insert (4.5) into (4.4a). This yields

$$\begin{aligned}
\mathbf{0} &= \widetilde{\mathbf{A}}_C\mathbf{C}\left(\widetilde{\mathbf{A}}_C^T\widetilde{\mathbf{e}},t\right)\widetilde{\mathbf{A}}_C^T\frac{d}{dt}\bar{\mathbf{e}} + \widetilde{\mathbf{A}}_C\mathbf{q}_t'\left(\widetilde{\mathbf{A}}_C^T\widetilde{\mathbf{e}},t\right) + \widetilde{\mathbf{A}}_R^T\mathbf{g}\left(\widetilde{\mathbf{A}}_R^T\widetilde{\mathbf{e}},t\right)\\
&\quad + \widetilde{\mathbf{A}}_L\mathbf{i}_L + \widetilde{\mathbf{A}}_V\mathbf{i}_V + \widetilde{\mathbf{A}}_{I,mod}\mathbf{i}_{mod}(*,t)
\end{aligned} \tag{4.6}$$

with

$$\bar{\mathbf{e}} := \begin{bmatrix} -\mathbf{F}_1^{-1}\mathbf{F}_2 \\ \mathbf{I} \end{bmatrix} \mathbf{e}_2,$$

$$\widetilde{\mathbf{A}}_{I,mod} := \begin{bmatrix} \widetilde{\mathbf{A}}_I \\ \widetilde{\mathbf{A}}_C \end{bmatrix},$$

$$\mathbf{i}_{mod}(*,t) := \begin{bmatrix} \mathbf{i}_s\left(\widetilde{\mathbf{A}}^T\widetilde{\mathbf{e}}, \frac{d}{dt}\mathbf{q}_C\left(\widetilde{\mathbf{A}}_C^T\widetilde{\mathbf{e}},t\right), \mathbf{i}_L, \mathbf{i}_V, t\right) \\ \mathbf{C}\left(\widetilde{\mathbf{A}}_C^T\widetilde{\mathbf{e}},t\right)\widetilde{\mathbf{A}}_C^T \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix} \mathbf{F}_1^{-1}\mathbf{Z}_{V-C}^T\frac{d}{dt}\mathbf{v}_{ind}(t) \end{bmatrix}.$$

Equation (4.6) has a structure similar to the structure of (3.22a). Since (4.5) is a solution for equation (4.4d), we can omit this equation and obtain a system of the same size as the original DAE (3.22).

4.2.2. Index reduction for circuits with LI cutsets

We consider a circuit that contains LI cutsets but no CV loops. Again, we have to examine those equations that yield hidden constraints and add them to (3.22). In this case the extended DAE is given by

$$\mathbf{0} = \mathbf{A}_C\frac{d}{dt}\mathbf{q}_C\left(\mathbf{A}_C^T\mathbf{e},t\right) + \mathbf{A}_R\mathbf{g}\left(\mathbf{A}_R^T\mathbf{e},t\right) + \mathbf{A}_L\mathbf{i}_L + \mathbf{A}_V\mathbf{i}_V + \mathbf{A}_I\mathbf{i}_s(*,t), \tag{4.7a}$$

$$\mathbf{0} = \frac{d}{dt}\mathbf{\Phi}_L(\mathbf{i}_L,t) - \mathbf{A}_L^T\mathbf{e}, \tag{4.7b}$$

$$\mathbf{0} = \mathbf{A}_V^T\mathbf{e} - \mathbf{v}_s(*,t), \tag{4.7c}$$

$$\mathbf{0} = \mathbf{Z}_{CRV}^T\mathbf{A}_L\frac{d}{dt}\mathbf{i}_L + \mathbf{Z}_{CRV}^T\mathbf{A}_I\frac{d}{dt}\mathbf{i}_s(*,t). \tag{4.7d}$$

Again, we will first try to obtain the derivatives (4.7d) in a way that allows for them to be easily inserted into the DAE.

Determination of the required derivatives

In the case of a circuit with LI cutsets, we will not determine $\mathbf{Z}_{CRV}^T\,[\mathbf{A}_L\ \mathbf{A}_I]$ as it was defined in Definition 3.41. Instead, we will use (3.27). Algorithm 3 is based on the duality of loops and cutsets that is described in Lemma 3.34. If we consider a tree in a graph, then we know by Definition 3.31 that every tree branch defines a fundamental cutset and every connecting branch defines a fundamental loop of the graph. Consider a connecting branch $\mathfrak{b}$ and the loop $\mathfrak{m}$ that is defined by $\mathfrak{b}$. The relations (3.8a) and (3.8b) show that $\mathfrak{b}$ belongs to every cutset that is defined by the tree branches in $\mathfrak{m}$. However, due to the minus sign in (3.8b), the orientations of $\mathfrak{b}$ in the cutsets is inverse to the orientations of the respective tree branches in the loop $\mathfrak{m}$.

Algorithm 3: Computation of $\mathbf{S}_{LI} = \mathbf{Z}_{CRV}^T [\mathbf{A}_I \ \mathbf{A}_L]$

```
Data: Network graph $\mathfrak{G}$
begin
1   determine the subgraph $\mathfrak{G}_{CRV}$ and $\mathfrak{G}_{-CRV}$ of $\mathfrak{G}$;
2   choose a tree of inductive branches $\mathfrak{T}_{L-CRV}$ in $\mathfrak{G}_{-CRV}$;
3   save the tree branches;
4   foreach connecting branch $\mathfrak{b}_c$ in $\mathfrak{G}_{-CRV}$ do
5       foreach tree branch $\mathfrak{b}_t$ in the loop $\mathfrak{m}$ defined by $\mathfrak{b}_c$ do
6           add $\mathfrak{b}_c$ to the list of branches in the cutset $\mathfrak{s}$ defined by $\mathfrak{b}_t$;
7   define the coefficients of $\mathbf{S}_{LI}$ by (3.7);
end
```

With Algorithm 3 the constraints are given by

$$\mathbf{0} = \mathbf{Z}_{CRV}^T [\mathbf{A}_I \ \mathbf{A}_L] \frac{d}{dt} \begin{bmatrix} \mathbf{i}_s(*,t) \\ \mathbf{i}_L \end{bmatrix}.$$

Moreover, if $\widehat{\mathbf{A}}_L$ is the incidence matrix of the inductive tree branches, then $\mathbf{Z}_{CRV}^T \widehat{\mathbf{A}}_L = \mathbf{I}_{N_{LI}}$ is nonsingular, cf. Lemma 3.34.

Algebraic transformation of the circuit DAE

We will use Algorithm 3 to determine the matrix $\mathbf{Z}_{CRV}^T [\mathbf{A}_L \ \mathbf{A}_I]$ as a fundamental LI cutset matrix and assume again that the branches are already ordered in such a way that $\mathbf{Z}_{CRV}^T [\mathbf{A}_L \ \mathbf{A}_I]$ can be written as in (3.28). In addition, we split $\mathbf{i}_L$ according to the splitting of $\mathbf{A}_L$ into $\mathbf{i}_L = \left[\widetilde{\mathbf{i}}_L^T \ \bar{\mathbf{i}}_L^T \ \widehat{\mathbf{i}}_L^T\right]^T$ and $i_s(*,t)$ according to the splitting of $\mathbf{A}_I$ in (3.29) into $i_s(*,t) = \left[i_{ind}(t)^T \ i_{contr}(*,t)^T\right]^T$. This allows us to rewrite (4.7d) as

$$\begin{aligned} \mathbf{0} &= \mathbf{Z}_{CRV}^T \mathbf{A}_L \frac{d}{dt}\mathbf{i}_L + \mathbf{Z}_{CRV}^T \mathbf{A}_I \frac{d}{dt}\mathbf{i}_s(*,t) \\ &= \mathbf{Z}_{CRV}^T \widetilde{\mathbf{A}}_L \frac{d}{dt}\widetilde{\mathbf{i}}_L + \frac{d}{dt}\widehat{\mathbf{i}}_L + \mathbf{Z}_{CRV}^T \mathbf{A}_{I,ind} \frac{d}{dt}\mathbf{i}_{ind}(t). \end{aligned} \tag{4.8}$$

Obviously, $\frac{d}{dt}\widehat{\mathbf{i}}_L$ is determined by $\frac{d}{dt}\widetilde{\mathbf{i}}_L$ and $\frac{d}{dt}\mathbf{i}_{ind}(t)$. Moreover, (4.8) can easily be solved for $\frac{d}{dt}\widehat{\mathbf{i}}_L$ and inserted into (4.7b). This leads to

$$\begin{aligned}\mathbf{0} &= \frac{d}{dt}\mathbf{\Phi}_L(\mathbf{i}_L, t) - \mathbf{A}_L^T e \\ &= \mathbf{L}(\mathbf{i}_L, t)\frac{d}{dt}\mathbf{i}_L + \mathbf{\Phi}_t'(\mathbf{i}_L, t) - \mathbf{A}_L^T\mathbf{e} \\ &= \mathbf{L}(\mathbf{i}_L, t)\begin{bmatrix} \frac{d}{dt}\widetilde{\mathbf{i}}_L \\ \frac{d}{dt}\bar{\mathbf{i}}_L \\ -\mathbf{Z}_{CRV}^T\widetilde{\mathbf{A}}_L\frac{d}{dt}\widetilde{\mathbf{i}}_L - \mathbf{Z}_{CRV}^T\mathbf{A}_{I,ind}\frac{d}{dt}\mathbf{i}_{ind}(t)\end{bmatrix} + \mathbf{\Phi}_t'(\mathbf{i}_L, t) - \mathbf{A}_L^T\mathbf{e} \qquad (4.9)\end{aligned}$$

If we replace equation (3.22b) in the DAE (3.22) by (4.9), then the resulting DAE has differentiation index 1.

If the circuit under consideration contains CV loops as well as LI cutsets, both approaches can be combined to obtain again a DAE system of differentiation index 1.

4.3. Index reduction for MNA c/f equations

In the previous section, we have adapted the index reduction by minimal extension method to DAEs that arise from MNA. We have used the special properties of such DAEs to obtain a differentiation index 1 system of the same size as the original differentiation index 2 system. If we want to apply a similar approach to DAEs from charge and flux oriented MNA, then we are faced with some difficulties in the case of constraints that arise from CV loops. This is because in contrast to DAEs from MNA the DAEs from charge and flux oriented MNA not only use node related values, i.e. the node potentials, but also branch related values, i.e. capacitance charges, to describe the behaviour of the capacitances. Hence, we have to find a way to resolve the interdependencies of both node related and branch related values. This will lead to an index reduction method that is based on the minimal extension method, but needs less derivatives of the charge conservation equations (3.21d) than the approach proposed in [48]. The same can be observed by examining the constraints that arise from LI cutsets. Again, we will aim at deriving a differentiation index 1 system that has the same size as the original differentiation index 2 system (3.21).

For the remainder of this section we will assume that the conditions in the following Assumption A2 hold. If this assumption does not hold, then the interdependencies of branch and node related values cannot be resolved correctly. This in turn means that we have to consider all equations in Theorem 3.43 in addition to the original DAE (3.21).

Assumption A2. *The capacitances and inductances of the circuit have to fulfill the following conditions.*

- *Any capacitance of the circuit depends only on the voltages across the element.*
- *Any inductance of the circuit depends only on the current through the element.*

- *Neither capacitances nor inductances are time dependent.*

This means that the capacitance and the inductance matrices have the forms

$$\mathbf{C}(\mathbf{v}_C,t) = \mathbf{C}(\mathbf{v}_C) = \begin{bmatrix} C_1(v_{C_1}) & & \\ & \ddots & \\ & & C_{N_C}(v_{C_{N_C}}) \end{bmatrix}, \tag{4.10a}$$

$$\mathbf{L}(\mathbf{i}_L,t) = \mathbf{L}(\mathbf{i}_L) = \begin{bmatrix} L_1(i_{L_1}) & & \\ & \ddots & \\ & & L_{N_L}(i_{L_{N_L}}) \end{bmatrix}. \tag{4.10b}$$

Remark 4.4. *Models for MOS transistors or MOSFETs that fulfill Assumption A2 violate the charge conservation for the respective transistor [33, 34]. To guarantee charge conservation, nonlinear capacitance models have been developed [70, 76, 78]. These models usually do not fulfill the Assumption A2. In this case, it is not possible to resolve the interdependencies between branch and node related values and an index reduction method would have to use the derivatives of the charge conservation laws associated with those transistors that are part of CV loops.*

As in the previous section, we will assume that Assumption A1 (cf. page 58) holds and discuss the case of CV loops and LI cutsets separately. Both transformations can be combined if the considered circuit contains both kinds of configurations that lead to differentiation index 2.

4.3.1. Index reduction for circuits with LI cutsets

We will begin with a circuit that contains LI cutsets but no CV loops and take a closer look at the LI cutset matrix $\mathbf{S}_{LI}$. According to Algorithm 3, $\mathbf{S}_{LI}$ can be computed in such a way that

$$\mathbf{S}_{LI} = \begin{bmatrix} \mathbf{S}_{I,ind} \; \mathbf{0}_{I,contr} \; \widetilde{\mathbf{S}}_L \; \mathbf{0}_L \; \mathbf{I}_{N_{LI}} \end{bmatrix}.$$

We will use the splittings for $\mathbf{A}_L$, $\mathbf{A}_I$, $\mathbf{i}_L$ and $\mathbf{i}_s(*,t)$ that we have defined in Section 4.2. By Assumption A2, we obtain for the flux conservation laws (3.21e)

$$\boldsymbol{\Phi} = \boldsymbol{\Phi}_L(\mathbf{i}_L,t) = \begin{bmatrix} \widetilde{\boldsymbol{\Phi}}_L(\widetilde{\mathbf{i}}_L) \\ \bar{\boldsymbol{\Phi}}_L(\bar{\mathbf{i}}_L) \\ \widehat{\boldsymbol{\Phi}}_L(\widehat{\mathbf{i}}_L) \end{bmatrix} = \begin{bmatrix} \widetilde{\boldsymbol{\Phi}}_L \\ \bar{\boldsymbol{\Phi}}_L \\ \widehat{\boldsymbol{\Phi}}_L \end{bmatrix}.$$

Furthermore, we define

$$\mathbf{L}(\mathbf{i}_L) = \frac{\partial}{\partial \mathbf{i}_L}\boldsymbol{\Phi}_L(\mathbf{i}_L) = \begin{bmatrix} \frac{\partial}{\partial \widetilde{\mathbf{i}}_L}\widetilde{\boldsymbol{\Phi}}_L(\widetilde{\mathbf{i}}_L) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{\partial}{\partial \bar{\mathbf{i}}_L}\bar{\boldsymbol{\Phi}}_L(\bar{\mathbf{i}}_L) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{\partial}{\partial \widehat{\mathbf{i}}_L}\widehat{\boldsymbol{\Phi}}_L(\widehat{\mathbf{i}}_L) \end{bmatrix}$$
$$=: \begin{bmatrix} \widetilde{\mathbf{L}}(\widetilde{\mathbf{i}}_L) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{L}}(\bar{\mathbf{i}}_L) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \widehat{\mathbf{L}}(\widehat{\mathbf{i}}_L) \end{bmatrix}.$$

With this splitting and with the identities given in (3.30), equation (3.26a) is transformed into

$$\mathbf{0} = \frac{d}{dt}\widehat{\mathbf{i}}_L + \mathbf{Z}_{CRV}^T\widetilde{\mathbf{A}}_L\frac{d}{dt}\widetilde{\mathbf{i}}_L + \mathbf{Z}_{CRV}^T\mathbf{A}_{I,ind}\frac{d}{dt}\mathbf{i}_{ind}(t) \tag{4.11}$$

and we see that only the derivatives of the branch currents through those inductances that are actually part of LI cutsets are of interest when reducing the differentiation index. To incorporate (4.11) into the system (3.21), we have to find a way to express those derivatives in a more suitable way. To do so, we take a look at the flux conservation law (3.21e). Differentiating the parts of (3.21e) that are associated with $\widehat{\boldsymbol{\Phi}}$ and $\widetilde{\boldsymbol{\Phi}}$ and taking (3.21b) into account yields

$$\widehat{\mathbf{A}}_L^T\mathbf{e} = \frac{d}{dt}\widehat{\boldsymbol{\Phi}} = \frac{\partial}{\partial \mathbf{i}}\widehat{\boldsymbol{\Phi}}_L(\widehat{\mathbf{i}}_L)\frac{d}{dt}\widehat{\mathbf{i}}_L =: \widehat{\mathbf{L}}(\widehat{\mathbf{i}}_L)\frac{d}{dt}\widehat{\mathbf{i}}_L, \tag{4.12a}$$
$$\widetilde{\mathbf{A}}_L^T\mathbf{e} = \frac{d}{dt}\widetilde{\boldsymbol{\Phi}} = \frac{\partial}{\partial \mathbf{i}}\widetilde{\boldsymbol{\Phi}}_L(\widetilde{\mathbf{i}}_L)\frac{d}{dt}\widetilde{\mathbf{i}}_L =: \widetilde{\mathbf{L}}(\widetilde{\mathbf{i}}_L)\frac{d}{dt}\widetilde{\mathbf{i}}_L, \tag{4.12b}$$

where, by Assumption A1, the Jacobians $\widehat{\mathbf{L}}(\mathbf{i})$ and $\widetilde{\mathbf{L}}(\mathbf{i})$ are positive definite. With (4.12) we can transform (4.11) as follows.

$$\begin{aligned}
&\mathbf{0} = \frac{d}{dt}\widehat{\mathbf{i}}_L + \mathbf{Z}_{CRV}^T\widetilde{\mathbf{A}}_L\frac{d}{dt}\widetilde{\mathbf{i}}_L + \mathbf{Z}_{CRV}^T\mathbf{A}_{I,ind}\frac{d}{dt}\mathbf{i}_{ind}(t)\\
\Longleftrightarrow &\mathbf{0} = \widehat{\mathbf{L}}(\widehat{\mathbf{i}}_L)\frac{d}{dt}\widehat{\mathbf{i}}_L + \widehat{\mathbf{L}}(\widehat{\mathbf{i}}_L)\left(\mathbf{Z}_{CRV}^T\widetilde{\mathbf{A}}_L\right)\frac{d}{dt}\widetilde{\mathbf{i}}_L + \widehat{\mathbf{L}}(\widehat{\mathbf{i}}_L)\left(\mathbf{Z}_{CRV}^T\mathbf{A}_{I,ind}\right)\frac{d}{dt}\mathbf{i}_{ind}(t)\\
\Longleftrightarrow &\mathbf{0} = \frac{d}{dt}\widehat{\boldsymbol{\Phi}} + \underbrace{\widehat{\mathbf{L}}(\widehat{\mathbf{i}}_L)\left(\mathbf{Z}_{CRV}^T\widetilde{\mathbf{A}}_L\right)\widetilde{\mathbf{L}}^{-1}(\widetilde{\mathbf{i}}_L)\widetilde{\mathbf{A}}_L^T\mathbf{e} + \widehat{\mathbf{L}}(\widehat{\mathbf{i}}_L)\left(\mathbf{Z}_{CRV}^T\mathbf{A}_{I,ind}\right)\frac{d}{dt}\mathbf{i}_{ind}(t)}_{=:-\mathcal{V}_L\left(\widehat{\mathbf{i}}_L,\widetilde{\mathbf{i}}_L,\widetilde{\mathbf{A}}_L^T\mathbf{e},t\right)}\\
\Longleftrightarrow &\frac{d}{dt}\widehat{\boldsymbol{\Phi}} = \mathcal{V}_L\left(\widehat{\mathbf{i}}_L,\widetilde{\mathbf{i}}_L,\widetilde{\mathbf{A}}_L^T\mathbf{e},t\right).
\end{aligned} \tag{4.13}$$

By inserting (4.13) into $\mathbf{0} = -\frac{d}{dt}\widehat{\mathbf{\Phi}} + \widehat{\mathbf{A}}_L^T\mathbf{e}$, we obtain the following DAE

$$\mathbf{0} = \mathbf{A}_C\frac{d}{dt}\mathbf{q} + \mathbf{A}_R\mathbf{g}\left(\mathbf{A}_R^T\mathbf{e}, t\right) + \widehat{\mathbf{A}}_L\widehat{\mathbf{i}}_L + \widetilde{\mathbf{A}}_L\widetilde{\mathbf{i}}_L + \bar{\mathbf{A}}_L\bar{\mathbf{i}}_L + \mathbf{A}_V\mathbf{i}_V + \mathbf{A}_I\mathbf{i}_s(*, t), \tag{4.14a}$$

$$\mathbf{0} = \widehat{\mathbf{A}}_L^T\mathbf{e} - \mathcal{V}_L\left(\widehat{\mathbf{i}}_L, \widetilde{\mathbf{i}}_L, \widetilde{\mathbf{A}}_L^T\mathbf{e}, t\right), \tag{4.14b}$$

$$\mathbf{0} = \frac{d}{dt}\widetilde{\mathbf{\Phi}} - \widetilde{\mathbf{A}}_L^T\mathbf{e}, \tag{4.14c}$$

$$\mathbf{0} = \frac{d}{dt}\bar{\mathbf{\Phi}} - \bar{\mathbf{A}}_L^T\mathbf{e}, \tag{4.14d}$$

$$\mathbf{0} = \mathbf{A}_V^T\mathbf{e} - \mathbf{v}_s(*, t), \tag{4.14e}$$

$$\mathbf{0} = \mathbf{q} - \mathbf{q}_C\left(\mathbf{A}_C^T\mathbf{e}\right), \tag{4.14f}$$

$$\mathbf{0} = \widehat{\mathbf{\Phi}} - \widehat{\mathbf{\Phi}}_L(\widehat{\mathbf{i}}_L), \tag{4.14g}$$

$$\mathbf{0} = \widetilde{\mathbf{\Phi}} - \widetilde{\mathbf{\Phi}}_L(\widetilde{\mathbf{i}}_L), \tag{4.14h}$$

$$\mathbf{0} = \bar{\mathbf{\Phi}} - \bar{\mathbf{\Phi}}_L(\bar{\mathbf{i}}_L). \tag{4.14i}$$

Theorem 4.5. *Consider a circuit that fulfills both Assumption A1 (p. 58) and Assumption A2 (p. 73). Moreover, assume that the circuit contains LI cutsets, but no CV loops. If the functions of the sources are sufficiently smooth and* $\widetilde{\mathbf{L}}(\widetilde{\mathbf{i}}_L)$ *and* $\widehat{\mathbf{L}}(\widehat{\mathbf{i}}_L)$ *are differentiable, then the DAE* (4.14) *has differentiation index 1.*

Proof. Equation (4.14b) can be interpreted as the constitutive laws for controlled voltage sources. From the splitting (3.28), we see that in each LI cutset one inductance is replaced by such a controlled source. From Theorem 4.3 we know that all other controlled sources do not influence the vector spaces which are critical for the index determination. Since we are only interested in the influence of the modified equation (4.14b) we will assume that the considered circuit only contains independent sources and write $i_s(t)$ and $v_s(t)$ in the following instead of $i_s(*, t)$ and $v_s(*, t)$.

Consider the matrices defined in Definition 3.41 and choose matrices $\mathbf{W}_C$, $\mathbf{W}_{V-C}$, and $\mathbf{W}_{R-CV}$ such that the matrices $[\mathbf{W}_* \ \mathbf{Z}_*]$, $* \in \{C, V-C, R-CV\}$ are nonsingular. Note that since we assumed that the circuit does not contain CV loops, $\bar{\mathbf{Z}}_{V-C}$ is void. Let

$$\mathbf{W} := \begin{bmatrix}\mathbf{W}_C & \mathbf{Z}_C\end{bmatrix}\begin{bmatrix}\mathbf{I}_C & \\ & \begin{bmatrix}\mathbf{W}_{V-C} & \mathbf{Z}_{V-C}\end{bmatrix}\end{bmatrix}\begin{bmatrix}\mathbf{I}_C & & \\ & \mathbf{I}_{V-C} & \\ & & \begin{bmatrix}\mathbf{W}_{R-CV} & \mathbf{Z}_{R-CV}\end{bmatrix}\end{bmatrix}, \tag{4.15}$$

where $\mathbf{I}_* \in \mathbb{R}^{m\times m}$ if $\mathbf{W}_* \in \mathbb{R}^{n\times m}$, $* \in \{C, V-C\}$, and define

$$\begin{bmatrix}\mathbf{W}_*^- \\ \mathbf{Z}_*^-\end{bmatrix} := [\mathbf{W}_* \ \mathbf{Z}_*]^{-1}, \ * \in \{C, V-C, R-CV\},$$

where $\mathbf{W}_*^- \in \mathbb{R}^{m\times n}$ and $\mathbf{Z}_*^- \in \mathbb{R}^{(n-m)\times n}$, if $\mathbf{W}_* \in \mathbb{R}^{n\times m}$ and $\mathbf{Z}_* \in \mathbb{R}^{n\times(n-m)}$. Hence we have

$$\mathbf{W}^{-1} = \begin{bmatrix} \mathbf{I}_C & & \\ & \mathbf{I}_{V-C} & \\ & & \begin{bmatrix} \mathbf{W}_{R-CV}^- \\ \mathbf{Z}_{R-CV}^- \end{bmatrix} \end{bmatrix} \begin{bmatrix} \mathbf{I}_C & \\ & \begin{bmatrix} \mathbf{W}_{V-C}^- \\ \mathbf{Z}_{V-C}^- \end{bmatrix} \end{bmatrix} \begin{bmatrix} \mathbf{W}_C^- \\ \mathbf{Z}_C^- \end{bmatrix}. \tag{4.16}$$

In addition, we set

$$\mathbf{W}^{-1}\mathbf{e} = \begin{bmatrix} \mathbf{W}_C^-\mathbf{e} \\ \mathbf{W}_{V-C}^-\mathbf{Z}_C^-\mathbf{e} \\ \mathbf{W}_{R-CV}^-\mathbf{Z}_{V-C}^-\mathbf{Z}_C^-\mathbf{e} \\ \mathbf{Z}_{CRV}^-\mathbf{e} \end{bmatrix} =: \begin{bmatrix} \mathbf{e}_C \\ \mathbf{e}_{V-C} \\ \mathbf{e}_{R-CV} \\ \mathbf{e}_{CRV} \end{bmatrix}, \tag{4.17a}$$

where $\mathbf{Z}_{CRV}^-$ denotes the product $\mathbf{Z}_{R-CV}^-\mathbf{Z}_{V-C}^-\mathbf{Z}_C^-$, and

$$\mathbf{W}_1 := \mathbf{W}_C, \qquad \mathbf{W}_2 := \mathbf{Z}_C\mathbf{W}_{V-C} \tag{4.17b}$$

$$\mathbf{W}_3 := \mathbf{Z}_C\mathbf{Z}_{V-C}\mathbf{W}_{R-CV}. \tag{4.17c}$$

We start by multiplying (4.14a) by $\mathbf{W}^T$ and applying (4.17). In this way, we obtain

$$\begin{aligned} \mathbf{0} = \mathbf{W}_1^T \Big[&\mathbf{A}_C\frac{d}{dt}\mathbf{q} + \widehat{\mathbf{A}}_L\widehat{\mathbf{i}}_L + \widetilde{\mathbf{A}}_L\widetilde{\mathbf{i}}_L + \bar{\mathbf{A}}_L\bar{\mathbf{i}}_L + \mathbf{A}_V\mathbf{i}_V + \mathbf{A}_I\mathbf{i}_s(t) \\ &+ \mathbf{A}_R\mathbf{g}\left(\mathbf{A}_R^T\left(\mathbf{W}_1\mathbf{e}_C + \mathbf{W}_2\mathbf{e}_{V-C} + \mathbf{W}_3\mathbf{e}_{R-CV}\right), t\right)\Big], \end{aligned} \tag{4.18a}$$

$$\begin{aligned} \mathbf{0} = \mathbf{W}_2^T \Big[&\widehat{\mathbf{A}}_L\widehat{\mathbf{i}}_L + \widetilde{\mathbf{A}}_L\widetilde{\mathbf{i}}_L + \bar{\mathbf{A}}_L\bar{\mathbf{i}}_L + \mathbf{A}_V\mathbf{i}_V + \mathbf{A}_I\mathbf{i}_s(t) \\ &+ \mathbf{A}_R\mathbf{g}\left(\mathbf{A}_R^T\left(\mathbf{W}_1\mathbf{e}_C + \mathbf{W}_2\mathbf{e}_{V-C} + \mathbf{W}_3\mathbf{e}_{R-CV}\right), t\right)\Big], \end{aligned} \tag{4.18b}$$

$$\begin{aligned} \mathbf{0} = \mathbf{W}_3^T \Big[&\widehat{\mathbf{A}}_L\widehat{\mathbf{i}}_L + \widetilde{\mathbf{A}}_L\widetilde{\mathbf{i}}_L + \bar{\mathbf{A}}_L\bar{\mathbf{i}}_L + \mathbf{A}_I\mathbf{i}_s(t) \\ &+ \mathbf{A}_R\mathbf{g}\left(\mathbf{A}_R^T\left(\mathbf{W}_1\mathbf{e}_C + \mathbf{W}_2\mathbf{e}_{V-C} + \mathbf{W}_3\mathbf{e}_{R-CV}\right), t\right)\Big] \end{aligned} \tag{4.18c}$$

and

$$\begin{aligned} \mathbf{0} &= \mathbf{Z}_{CRV}^T\left[\widehat{\mathbf{A}}_L\widehat{\mathbf{i}}_L + \widetilde{\mathbf{A}}_L\widetilde{\mathbf{i}}_L + \bar{\mathbf{A}}_L\bar{\mathbf{i}}_L + \mathbf{A}_I\mathbf{i}_s(*, t)\right] \\ &= \widehat{\mathbf{i}}_L + \mathbf{Z}_{CRV}^T\widetilde{\mathbf{A}}_L\widetilde{\mathbf{i}}_L + \mathbf{Z}_{CRV}^T\mathbf{A}_I\mathbf{i}_s(t). \end{aligned} \tag{4.18d}$$

The same transformation of $\mathbf{e}$ is applied to (4.14b)-(4.14f), so that we obtain

$$\begin{aligned}
\mathbf{0} &= \widehat{\mathbf{A}}_L^T\left(\mathbf{W}_1\mathbf{e}_C + \mathbf{W}_2\mathbf{e}_{V-C} + \mathbf{W}_3\mathbf{e}_{R-CV}\right) + \mathbf{e}_{CRV} - \mathcal{V}_L\left(\widehat{\mathbf{i}}_L, \widetilde{\mathbf{i}}_L, \widetilde{\mathbf{A}}_L^T\mathbf{e}, t\right) \\
&= \left(\mathbf{I}_{N_{LI}} + \widehat{\mathbf{L}}(\widehat{\mathbf{i}}_L)\mathbf{Z}_{CRV}^T\widetilde{\mathbf{A}}_L\widetilde{\mathbf{L}}^{-1}(\widetilde{\mathbf{i}}_L)\widetilde{\mathbf{A}}_L^T\mathbf{Z}_{CRV}\right)\mathbf{e}_{CRV} + \widehat{\mathbf{L}}(\widehat{\mathbf{i}}_L)\mathbf{Z}_{CRV}^T\mathbf{A}_I\frac{d}{dt}\mathbf{i}_s(t) \\
&\quad + \left(\widehat{\mathbf{A}}_L + \widehat{\mathbf{L}}(\widehat{\mathbf{i}}_L)\mathbf{Z}_{CRV}^T\widetilde{\mathbf{A}}_L\widetilde{\mathbf{L}}^{-1}(\widetilde{\mathbf{i}}_L)\right)\left(\mathbf{W}_1\mathbf{e}_C + \mathbf{W}_2\mathbf{e}_{V-C} + \mathbf{W}_3\mathbf{e}_{R-CV}\right), \qquad (4.19a) \\
\mathbf{0} &= \frac{d}{dt}\widetilde{\mathbf{\Phi}} - \widetilde{\mathbf{A}}_L^T\left(\mathbf{W}_1\mathbf{e}_C + \mathbf{W}_2\mathbf{e}_{V-C} + \mathbf{W}_3\mathbf{e}_{R-CV} + \mathbf{Z}_{CRV}\mathbf{e}_{CRV}\right), \qquad (4.19b) \\
\mathbf{0} &= \frac{d}{dt}\bar{\mathbf{\Phi}} - \bar{\mathbf{A}}_L^T\left(\mathbf{W}_1\mathbf{e}_C + \mathbf{W}_2\mathbf{e}_{V-C} + \mathbf{W}_3\mathbf{e}_{R-CV}\right), \qquad (4.19c) \\
\mathbf{0} &= \mathbf{A}_V^T\left(\mathbf{W}_1\mathbf{e}_C + \mathbf{W}_2\mathbf{e}_{V-C}\right) - \mathbf{v}_s(t) \qquad (4.19d)
\end{aligned}$$

and

$$\mathbf{0} = \mathbf{q} - \mathbf{q}_C(\mathbf{A}_C^T\mathbf{W}_1\mathbf{e}_C). \qquad (4.19e)$$

The flux equations (4.14g)-(4.14i) remain unchanged. Now, we differentiate (4.19e) which yields

$$\frac{d}{dt}\mathbf{q} = \mathbf{C}\left(\mathbf{A}_C^T\mathbf{W}_1\mathbf{e}_C\right)\mathbf{A}_C^T\mathbf{W}_1\frac{d}{dt}\mathbf{e}_C, \qquad (4.20)$$

where $\mathbf{C}(\mathbf{v}_C) = \frac{\partial}{\partial \mathbf{v}_C}\mathbf{q}_C(\mathbf{v}_C)$ is again the capacitance matrix. Equation (4.20) provides an expression for $\frac{d}{dt}\mathbf{q}$ which we insert into (4.18a) to obtain

$$\begin{aligned}
\mathbf{W}_1^T\mathbf{A}_C\mathbf{C}(\mathbf{A}_C^T\mathbf{W}_1\mathbf{e}_C)\mathbf{A}_C^T\mathbf{W}_1\frac{d}{dt}\mathbf{e}_C = -\mathbf{W}_1^T\Big[&\widehat{\mathbf{A}}_L\widehat{\mathbf{i}}_L + \widetilde{\mathbf{A}}_L\widetilde{\mathbf{i}}_L + \bar{\mathbf{A}}_L\bar{\mathbf{i}}_L + \mathbf{A}_V\mathbf{i}_V + \mathbf{A}_I\mathbf{i}_s(t) \\
&+\mathbf{A}_R\mathbf{g}\left(\mathbf{A}_R^T\mathbf{e}, t\right)\Big].
\end{aligned} \qquad (4.21)$$

The matrix $\mathbf{W}_1^T\mathbf{A}_C\mathbf{C}(\mathbf{A}_C^T\mathbf{W}_1\mathbf{e}_C)\mathbf{A}_C^T\mathbf{W}_1$ is nonsingular because $\mathbf{C}(\mathbf{v}_C)$ is positive definite by Assumption A1 and $\mathbf{W}_1\mathbf{A}_C^T$ has full column rank. Thus, we have found an expression for $\frac{d}{dt}\mathbf{e}_C$. The derivative of (4.19d) is given by

$$\mathbf{A}_V^T\mathbf{W}_2\frac{d}{dt}\mathbf{e}_{V-C} = -\mathbf{A}_V^T\mathbf{W}_1\frac{d}{dt}\mathbf{e}_C + \frac{d}{dt}\mathbf{v}_s(t) \qquad (4.22)$$

and we can use (4.21) to replace $\frac{d}{dt}\mathbf{e}_C$. Since $\mathbf{A}_V^T\mathbf{W}_2 = \mathbf{A}_V^T\mathbf{Z}_C\mathbf{W}_{V-C}$ is nonsingular, (4.22) leads to a differential equation for $\frac{d}{dt}\mathbf{e}_{V-C}$. We differentiate (4.14g), (4.14h) and (4.14i) to obtain

$$\begin{aligned}
\frac{d}{dt}\widehat{\mathbf{\Phi}} &= \widehat{\mathbf{L}}(\widehat{\mathbf{i}}_L)\frac{d}{dt}\widehat{\mathbf{i}}_L, \qquad (4.23a) \\
\frac{d}{dt}\widetilde{\mathbf{\Phi}} &= \widetilde{\mathbf{L}}(\widetilde{\mathbf{i}}_L)\frac{d}{dt}\widetilde{\mathbf{i}}_L, \qquad (4.23b) \\
\frac{d}{dt}\bar{\mathbf{\Phi}} &= \bar{\mathbf{L}}(\bar{\mathbf{i}}_L)\frac{d}{dt}\bar{\mathbf{i}}_L. \qquad (4.23c)
\end{aligned}$$

These expressions for the derivatives of the fluxes are inserted into (4.19b) and (4.19c) which leads to

$$\widetilde{\mathbf{L}}(\widetilde{\mathbf{i}}_L)\frac{d}{dt}\widetilde{\mathbf{i}}_L = \widetilde{\mathbf{A}}_L^T\left(\mathbf{W}_1\mathbf{e}_C + \mathbf{W}_2\mathbf{e}_{V-C} + \mathbf{W}_3\mathbf{e}_{R-CV} + \mathbf{Z}_{CRV}\mathbf{e}_{CRV}\right), \tag{4.24a}$$

$$\bar{\mathbf{L}}(\bar{\mathbf{i}}_L)\frac{d}{dt}\bar{\mathbf{i}}_L = \bar{\mathbf{A}}_L^T\left(\mathbf{W}_1\mathbf{e}_C + \mathbf{W}_2\mathbf{e}_{V-C} + \mathbf{W}_3\mathbf{e}_{R-CV}\right). \tag{4.24b}$$

Since both $\widetilde{\mathbf{L}}(\widetilde{\mathbf{i}}_L)$ and $\bar{\mathbf{L}}(\bar{\mathbf{i}}_L)$ are positive definite by Assumption A1, equations (4.24a) and (4.24b) yield differential equations for $\frac{d}{dt}\widetilde{\mathbf{i}}_L$ and $\frac{d}{dt}\bar{\mathbf{i}}_L$. The differential equation for $\frac{d}{dt}\widehat{\mathbf{i}}_L$ is derived by differentiating (4.18d) and replacing $\frac{d}{dt}\widetilde{\mathbf{i}}_L$ with help of (4.24a). It is given by

$$\frac{d}{dt}\widehat{\mathbf{i}}_L = -\mathbf{Z}_{CRV}^T\left[\widetilde{\mathbf{A}}_L\widetilde{\mathbf{L}}^{-1}(\widetilde{\mathbf{i}}_L)\widetilde{\mathbf{A}}_L^T\left(\mathbf{W}_1\mathbf{e}_C + \mathbf{W}_2\mathbf{e}_{V-C} + \mathbf{W}_3\mathbf{e}_{R-CV} + \mathbf{Z}_{CRV}\mathbf{e}_{CRV}\right)\right.$$
$$\left. + \mathbf{A}_I\frac{d}{dt}\mathbf{i}_s(t)\right]. \tag{4.25}$$

The derivative of (4.18c) is given by

$$\begin{aligned}\mathbf{W}_3^T\mathbf{A}_R\mathbf{G}(\mathbf{A}_R^T\mathbf{e},t)\mathbf{A}_R^T\mathbf{W}_3\frac{d}{dt}\mathbf{e}_{R-CV} = -\mathbf{W}_3^T\Bigg[&\widehat{\mathbf{A}}_L\frac{d}{dt}\widehat{\mathbf{i}}_L + \widetilde{\mathbf{A}}_L\frac{d}{dt}\widetilde{\mathbf{i}}_L + \bar{\mathbf{A}}_L\frac{d}{dt}\bar{\mathbf{i}}_L + \mathbf{A}_I\frac{d}{dt}\mathbf{i}_s(t)\\ &+ \mathbf{G}\left(\mathbf{A}_R^T\mathbf{e},t\right)\mathbf{A}_R^T\left(\mathbf{W}_1\frac{d}{dt}\mathbf{e}_C + \mathbf{W}_2\frac{d}{dt}\mathbf{e}_{V-C}\right)\\ &+ \mathbf{g}_t'\left(\mathbf{A}_R^T\mathbf{e},t\right)\Bigg],\end{aligned} \tag{4.26}$$

where $\mathbf{G}(\mathbf{v}_R,t) = \frac{\partial}{\partial\mathbf{v}_R}\mathbf{g}(\mathbf{v}_R,t)$ is the conductance matrix and $\mathbf{g}_t'(\mathbf{v}_R,t)$ as defined in (3.18). We can use equations (4.21), (4.22), (4.24a), (4.24b) and (4.25) to eliminate all differential variables except $\frac{d}{dt}\mathbf{e}_{R-CV}$. The matrix $\mathbf{W}_3^T\mathbf{A}_R\mathbf{G}(\mathbf{A}_R^T\mathbf{e},t)\mathbf{A}_R^T\mathbf{W}_3$ is again nonsingular, since $\mathbf{G}(\mathbf{v}_R,t)$ is positive definite and $\mathbf{A}_R^T\mathbf{W}_3 = \mathbf{A}_R^T\mathbf{Z}_C\mathbf{Z}_{V-C}\mathbf{W}_{R-CV}$ has full column rank. Hence, (4.26) provides a differential equation for $\frac{d}{dt}\mathbf{e}_{R-CV}$. Next, we differentiate (4.18b) and obtain

$$\begin{aligned}\mathbf{W}_2^T\mathbf{A}_V\frac{d}{dt}\mathbf{i}_V = -\mathbf{W}_2^T\Bigg[&\widehat{\mathbf{A}}_L\frac{d}{dt}\widehat{\mathbf{i}}_L + \widetilde{\mathbf{A}}_L\frac{d}{dt}\widetilde{\mathbf{i}}_L + \bar{\mathbf{A}}_L\frac{d}{dt}\bar{\mathbf{i}}_L + \mathbf{A}_I\frac{d}{dt}\mathbf{i}_s(t) + \mathbf{A}_R\mathbf{g}_t'(\mathbf{A}_R^T\mathbf{e},t)\\ &+\mathbf{A}_R\mathbf{G}(\mathbf{A}_R^T\mathbf{e},t)\mathbf{A}_R^T\left(\mathbf{W}_1\frac{d}{dt}\mathbf{e}_C + \mathbf{W}_2\frac{d}{dt}\mathbf{e}_{V-C} + \mathbf{W}_3^T\frac{d}{dt}\mathbf{e}_{R-CV}\right)\Bigg].\end{aligned} \tag{4.27}$$

Again, we are able to replace all differential variables except $\frac{d}{dt}\mathbf{i}_V$. The matrix $\mathbf{W}_2^T\mathbf{A}_V = \mathbf{W}_{V-C}^T\mathbf{Z}_C^T\mathbf{A}_V$ is nonsingular, since $\mathbf{A}_V$ has full column rank. Thus, (4.27) yields a differential equation for $\frac{d}{dt}\mathbf{i}_V$. Finally, we differentiate (4.19a) and set

$$\begin{aligned}
\mathbf{T}_1(\widehat{\mathbf{i}}_L,\widetilde{\mathbf{i}}_L) :=& \mathbf{I}_{N_{LI}} + \widehat{\mathbf{L}}(\widehat{\mathbf{i}}_L)\mathbf{Z}_{CRV}^T\widetilde{\mathbf{A}}_L\widetilde{\mathbf{L}}^{-1}(\widetilde{\mathbf{i}}_L)\widetilde{\mathbf{A}}_L^T\mathbf{Z}_{CRV},\\
\mathbf{T}_2(\widehat{\mathbf{i}}_L,\widetilde{\mathbf{i}}_L) :=& \widehat{\mathbf{A}}_L + \widehat{\mathbf{L}}(\widehat{\mathbf{i}}_L)\mathbf{Z}_{CRV}^T\widetilde{\mathbf{A}}_L\widetilde{\mathbf{L}}^{-1}(\widetilde{\mathbf{i}}_L)\\
\mathcal{T}_k :=& \left[\frac{\partial}{\partial\widehat{\mathbf{i}}_L}\mathbf{T}_k(\widehat{\mathbf{i}}_L,\widetilde{\mathbf{i}}_L)\frac{d}{dt}\widehat{\mathbf{i}}_L + \frac{\partial}{\partial\widetilde{\mathbf{i}}_L}\mathbf{T}_k(\widehat{\mathbf{i}}_L,\widetilde{\mathbf{i}}_L)\frac{d}{dt}\widetilde{\mathbf{i}}_L\right],\ k=1,2
\end{aligned}$$

and get

$$\begin{aligned}
\mathbf{T}_1(\widehat{\mathbf{i}}_L,\widetilde{\mathbf{i}}_L)\frac{d}{dt}\mathbf{e}_{CRV} =& -\mathcal{T}_1\mathbf{e}_{CRV} - \mathcal{T}_2\left(\mathbf{W}_1\mathbf{e}_C + \mathbf{W}_2\mathbf{e}_{V-C} + \mathbf{W}_3\mathbf{e}_{R-CV}\right)\\
&- \mathbf{T}_2(\widehat{\mathbf{i}}_L,\widetilde{\mathbf{i}}_L)\left(\mathbf{W}_1\frac{d}{dt}\mathbf{e}_C + \mathbf{W}_2\frac{d}{dt}\mathbf{e}_{V-C} + \mathbf{W}_3\frac{d}{dt}\mathbf{e}_{R-CV}\right)\\
&- \widehat{\mathbf{L}}(\widehat{\mathbf{i}}_L)\mathbf{Z}_{CRV}^T\widetilde{\mathbf{A}}_I\frac{d^2}{dt^2}\mathbf{i}_s(t) - \left[\frac{\partial}{\partial\widehat{\mathbf{i}}_L}\widehat{\mathbf{L}}(\widehat{\mathbf{i}}_L)\frac{d}{dt}\widehat{\mathbf{i}}_L\right]\frac{d}{dt}\mathbf{i}_s(t).
\end{aligned}\tag{4.28}$$

The differential variables $\frac{d}{dt}\widehat{\mathbf{i}}_L$, $\frac{d}{dt}\widetilde{\mathbf{i}}_L$, $\frac{d}{dt}\mathbf{e}_C$, $\frac{d}{dt}\mathbf{e}_{V-C}$ and $\frac{d}{dt}\mathbf{e}_{R-CV}$ can be eliminated with the help of the previously derived differential equations. Since $\widehat{\mathbf{L}}(\widehat{\mathbf{i}}_L)$ is positive definite and $\mathbf{Z}_{CRV}^T\widetilde{\mathbf{A}}_L\widetilde{\mathbf{L}}^{-1}(\widetilde{\mathbf{i}}_L)\widetilde{\mathbf{A}}_L^T\mathbf{Z}_{CRV}$ is at least positive semidefinite, $\mathbf{T}_1$ is nonsingular and (4.28) is a differential equation for $\frac{d}{dt}\mathbf{e}_{CRV}$.

To conclude the proof, we note that in order to derive the ODE system (4.20)-(4.28), we had to differentiate every equation at most one time. Hence, (4.14) has differentiation index 1. □

Proposition 4.6. *The DAE* (4.14) *has a properly stated leading term.*

Proof. We rewrite (4.14) as

$$\begin{aligned}
\mathbf{0} &= \mathbf{A}_C\frac{d}{dt}\mathbf{q}_C\left(\mathbf{A}_C^T\mathbf{e}\right) + \mathbf{A}_R\mathbf{g}\left(\mathbf{A}_R^T\mathbf{e},t\right) + \widehat{\mathbf{A}}_L\widehat{\mathbf{i}}_L + \widetilde{\mathbf{A}}_L\widetilde{\mathbf{i}}_L + \bar{\mathbf{A}}_L\bar{\mathbf{i}}_L + \mathbf{A}_V\mathbf{i}_V + \mathbf{A}_I\mathbf{i}_s(*,t),\\
\mathbf{0} &= \frac{d}{dt}\widetilde{\mathbf{\Phi}}_L(\widetilde{\mathbf{i}}_L) - \widetilde{\mathbf{A}}_L^T\mathbf{e},\\
\mathbf{0} &= \frac{d}{dt}\bar{\mathbf{\Phi}}_L(\bar{\mathbf{i}}_L) - \bar{\mathbf{A}}_L^T\mathbf{e},\\
\mathbf{0} &= \widehat{\mathbf{A}}_L^T\mathbf{e} - \mathcal{V}_L\left(\widehat{\mathbf{i}}_L,\widetilde{\mathbf{i}}_L,\widetilde{\mathbf{A}}_L^T\mathbf{e},t\right),\\
\mathbf{0} &= \mathbf{A}_V^T\mathbf{e} - \mathbf{v}_s(*,t),
\end{aligned}\tag{4.29}$$

which is the MNA formulation of (4.14). In analogy to [59], we choose

$$\mathbf{A} = \begin{bmatrix}\mathbf{A}_C & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}_{\widetilde{L}} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{\bar{L}}\\ \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix},\quad \mathbf{R} = \begin{bmatrix}\mathbf{A}_C^-\mathbf{A}_C & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}_{\widetilde{L}} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{\bar{L}}\end{bmatrix}\quad\text{and}\quad \mathbf{d}(t,\mathbf{x}) = \begin{bmatrix}\mathbf{q}_C\left(\mathbf{A}_C^T\mathbf{e}\right)\\ \widetilde{\mathbf{\Phi}}_L(\widetilde{\mathbf{i}}_L)\\ \bar{\mathbf{\Phi}}_L(\bar{\mathbf{i}}_L)\end{bmatrix},$$

where $\mathbf{A}_C^-$ denotes the Moore-Penrose inverse of $\mathbf{A}_C$ and $\mathbf{x} = \left[\mathbf{e}^T, \widehat{\mathbf{i}}_L^T, \widetilde{\mathbf{i}}_L^T, \bar{\mathbf{i}}_L^T, \mathbf{i}_V^T\right]^T$. By construction, we have that

$$\ker \mathbf{A} = \ker \mathbf{R}.$$

Moreover, we have

$$\mathbf{D}(t, \mathbf{x}) = \frac{\partial}{\partial \mathbf{x}} \mathbf{d}(t, \mathbf{x}) = \begin{bmatrix} \mathbf{C}\left(\mathbf{A}_C^T \mathbf{e}\right) \mathbf{A}_C^T & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \widetilde{\mathbf{L}}\left(\widetilde{\mathbf{i}}_L\right) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \bar{\mathbf{L}}\left(\bar{\mathbf{i}}_L\right) & \mathbf{0} \end{bmatrix},$$

with $\mathbf{C}\left(\mathbf{v}_C\right)$, $\widetilde{\mathbf{L}}\left(\widetilde{\mathbf{i}}_L\right)$ and $\bar{\mathbf{L}}\left(\bar{\mathbf{i}}_L\right)$ fulfilling Assumptions A1 and A2. Hence, we also have

$$\operatorname{range} \mathbf{D}(t, \mathbf{x}) = \operatorname{range} \mathbf{R}$$

and we see that (4.29) has a properly stated leading term. According to [59], this also shows that (4.14) has properly stated leading term. □

4.3.2. Index reduction for circuits with CV loops

In this section, we will consider a circuit that contains CV loops but no LI cutsets. Whereas the index reduction in case of LI cutsets could be easily transferred from the MNA case to the MNA c/f case, we are faced with some problems in the case of CV loops. In the case of LI cutsets we only had to deal with values that were related to inductive branches, namely the inductive branch currents $\mathbf{i}_L$ and the inductive fluxes $\mathbf{\Phi}$. However, if we consider the constraints that arise from CV loops for the MNA c/f, we have on one hand (3.24a)

$$\mathbf{0} = \bar{\mathbf{Z}}_{V-C}^T \Big(\mathbf{A}_V^T \frac{d}{dt} \mathbf{e} - \frac{d}{dt} \mathbf{v}_s(*, t) \Big),$$

but on the other hand (3.24b)

$$\mathbf{0} = \frac{d}{dt} \mathbf{q} - \frac{d}{dt} \mathbf{q}_C(\mathbf{A}_C^T \mathbf{e}, t).$$

These equations establish a connection between branch related values, namely the capacitive charges $\mathbf{q}$, and node related values, namely the node potentials $\mathbf{e}$. Moreover, the equations are connected to different types of elements, namely capacitances and voltage sources. Hence, our first task will be to find a different representation for the constraints (3.24) that resolves these connections.

Determination of required derivatives

To find a suitable representation of the constraints, we will first reformulate (3.24a) in such a way that the new equations not only involve the voltage sources that are part of CV loops, but also the respective capacitances.

We start by taking a closer look at (3.24a). In Section 3.3.2, we have shown that the matrix $\bar{\mathbf{Z}}_{V-C}^T$ basically is a fundamental loop matrix of the V loops in the subgraph $\mathfrak{G}_{V-C}$. In order to combine (3.24a) and (3.24b), we have to include the capacitances into our considerations. Note that the nodes of $\mathfrak{G}_{V-C}$ are either nodes of the network graph $\mathfrak{G}$ which are not incident with capacitances or the connected components of the capacitive subgraph $\mathfrak{G}_C$. Hence, we have to extend the V loops to CV loops in $\mathfrak{G}_{V-C}$. However, the extension of a V loop to a CV loop may not be unique.

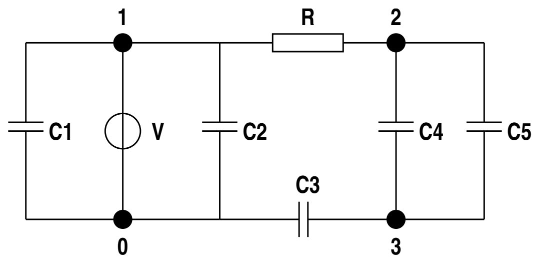


Figure 4.1.: Circuit with two CV loops

Example 4.7. Consider the circuit in Figure 4.1. After contracting the C-subgraph $\mathfrak{G}_C$, we find the V loop that consists of the voltage source V. If we want to extend the V loop to a CV loop, we have two alternatives to do so, namely either the loop C_1, V or the loop C_2, V. Note that if we choose a tree in the network graph that includes the voltage source V, then both loops are fundamental loops and hence independent. Note also that the capacitances C_1 and C_2 form a C loop as do the capacitances C_4 and C_5. Whereas the capacitances in the first C loop are part of CV loops, the capacitances of the latter C loop are not part of any CV loop. □

Example 4.7 shows that we are able to find a set of independent CV loops by examining the connected components of the subgraph $\mathfrak{G}_{CV}$ of the network graph. However, we have to find a way to distinguish C loops in which all capacitances are part of CV loops and C loops in which no capacitance is part of a CV loop. Now, we know from Lemma 3.20, that C and CV loops can only occur in the blocks of the subgraph $\mathfrak{G}_{CV}$. Moreover, if a block of $\mathfrak{G}_{CV}$ contains at least one voltage source V, then for each capacitance C in the same block we are able to find a CV loop that contains both V and C. Hence, we need to determine the blocks of $\mathfrak{G}_{CV}$ and check each block for voltage sources. In both [41] and [71], algorithms for the determination of blocks of a non-oriented graph are given. We can apply these algorithms here, as at this point we are only interested in whether loops exist or not and not in the special directions of branches inside the loops. Algorithm 4 is taken from [41] and adapted to the problem at hand. The sets B_k determined by Algorithm 4 are the sets of nodes in the blocks of $\mathfrak{G}_{CV}$. We can define a block $\mathfrak{K}_k$ by the nodes in B_k and all branches that are incident with a pair of nodes in B_k.

Now, consider a block $\mathfrak{K}$ of $\mathfrak{G}_{CV}$ that contains a voltage source. In order to determine a set of fundamental loops in $\mathfrak{K}$, we first have to compute a tree in $\mathfrak{K}$. This will be done in such a way that the tree includes all voltage sources. Algorithm 5 then determines a fundamental CV loop matrix of the following form

$$\mathbf{M}_{CV} = \Big[\; \underbrace{\mathbf{I}_{\widetilde{N}_{CV}}}_{\substack{\text{capacitive}\\ \text{connecting}\\ \text{branches}\\ \text{inside}\\ \text{CV loops}}} \quad \underbrace{\widetilde{\mathbf{M}}_C}_{\substack{\text{capacitive}\\ \text{tree}\\ \text{branches}\\ \text{inside}\\ \text{CV loops}}} \quad \underbrace{\mathbf{0}_C}_{\substack{\text{capacitive}\\ \text{branches}\\ \text{outside}\\ \text{CV loops}}} \quad \underbrace{\mathbf{M}_{V,ind}}_{\substack{\text{independent}\\ \text{voltage}\\ \text{sources}}} \quad \underbrace{\mathbf{0}_{V,contr}}_{\substack{\text{controlled}\\ \text{voltage}\\ \text{sources}}} \Big], \tag{4.30}$$

where $\widetilde{N}_{CV}$ is the number of fundamental CV loops in $\mathfrak{G}_{CV}$. In order to derive the equations which are necessary to perform an index reduction, we apply Kirchhoff's Voltage Law to these fundamental CV loops. Let the vectors $\mathbf{v}_C$ and $\mathbf{v}_V$ be ordered such that

$$\mathbf{A}_C^T\mathbf{e} = \mathbf{v}_C =: \begin{bmatrix} \widehat{\mathbf{v}}_C \\ \widetilde{\mathbf{v}}_C \\ \bar{\mathbf{v}}_C \end{bmatrix} =: \begin{bmatrix} \widehat{\mathbf{A}}_C^T\mathbf{e} \\ \widetilde{\mathbf{A}}_C^T\mathbf{e} \\ \bar{\mathbf{A}}_C^T\mathbf{e} \end{bmatrix} \tag{4.31}$$

and

$$\mathbf{v}_V = \begin{bmatrix} \mathbf{v}_{V,ind} \\ \mathbf{v}_{V,contr} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_{ind}(t) \\ \mathbf{v}_{contr}(*,t) \end{bmatrix}.$$

Then Kirchoff's Voltage Laws yields

$$\mathbf{0} = \mathbf{M}_{CV} \begin{bmatrix} \mathbf{v}_C \\ \mathbf{v}_V \end{bmatrix} = \widehat{\mathbf{v}}_C + \widetilde{\mathbf{M}}_C\mathbf{v}_C + \mathbf{M}_{V,ind}\mathbf{v}_{ind}(t)$$

or if we consider the derivative

$$\mathbf{0} = \frac{d}{dt}\widehat{\mathbf{v}}_C + \widetilde{\mathbf{M}}_C\frac{d}{dt}\widetilde{\mathbf{v}}_C + \mathbf{M}_{V,ind}\frac{d}{dt}\mathbf{v}_{ind}(t). \tag{4.32}$$

Equation (4.32) is equivalent to the original derivative (3.24a) since KVL also yields

$$\frac{d}{dt}\widehat{\mathbf{v}}_C + \widetilde{\mathbf{M}}_C\frac{d}{dt}\widetilde{\mathbf{v}}_C = -\mathbf{M}_{V,ind}\frac{d}{dt}\mathbf{v}_{V,ind} = -\mathbf{M}_{V,ind}\mathbf{A}_V^T\frac{d}{dt}\mathbf{e}$$

which can be inserted into (4.32). Moreover, since we included all voltage sources that belong to CV loops in our considerations and since we determined a set of fundamental CV loops there exists a nonsingular matrix $\mathbf{T}$ such that

$$\mathbf{T}\mathbf{M}_V = \begin{bmatrix} \bar{\mathbf{Z}}_{V-C}^T \\ \mathbf{0} \end{bmatrix}.$$

Algorithm 4: Computation of the blocks of $\mathfrak{G}_{CV}$

```
Data: Subgraph $\mathfrak{G}_{CV}(\mathfrak{N}_{CV}, \mathfrak{B}_{CV})$ and starting node $\mathfrak{n}^* \in \mathfrak{N}_{CV}$
begin
1    foreach $\mathfrak{n} \in \mathfrak{N}_{CV}$ do $nr(\mathfrak{n}) = 0$; $p(\mathfrak{n}) = 0$;
2    foreach $\mathfrak{b} \in \mathfrak{B}_{CV}$ do $u(\mathfrak{b}) = \mathsf{false}$;
3    $i = 0$; $\mathfrak{n} = \mathfrak{n}^*$; $C = \emptyset$; $k = 0$; $L(\mathfrak{n}^*) = 0$; create a stack $S$ with single element $\mathfrak{n}^*$;
end
repeat
     while there exists a branch $\mathfrak{b} =< \mathfrak{n}, \mathfrak{n}' >$ with $u(\mathfrak{b}) = \mathsf{false}$ do
4        choose $\mathfrak{n}'$; $u(\mathfrak{b}) = \mathsf{true}$;
         if $nr(\mathfrak{n}') = 0$ then
5            $p(\mathfrak{n}') = \mathfrak{n}$; $i = i + 1$; $nr(\mathfrak{n}') = i$; $L(\mathfrak{n}') = i$; append $\mathfrak{n}'$ to $S$; $\mathfrak{n} = \mathfrak{n}'$;
6        else $L(\mathfrak{n}) = \min\{L(\mathfrak{n}), nr(\mathfrak{n}')\}$;
     if $p(\mathfrak{n}) \neq \mathfrak{n}^*$ then
7        if $L(\mathfrak{n}) < nr(p(\mathfrak{n}))$ then $L(p(\mathfrak{n})) = \min\{L(p(\mathfrak{n})), L(\mathfrak{n})\}$;
         else
8            $C = C \cup \{p(\mathfrak{n})\}$; $k = k + 1$;
9            create a list $B_k$ containing all nodes of $S$ up to $\mathfrak{n}$ (including $\mathfrak{n}$); remove
             these nodes from $S$; append $p(\mathfrak{n})$ to $B_k$;
     else
10       if there exists a branch $\mathfrak{b} =< \mathfrak{n}^*, \mathfrak{n}' >$ with $u(\mathfrak{b}) = \mathsf{false}$ then $C = C \cup \{\mathfrak{n}^*\}$;
11       $k = k + 1$; create a list $B_k$ containing all nodes of $S$ up to $\mathfrak{n}$ (including $\mathfrak{n}$);
         remove these nodes from $S$; append $p(\mathfrak{n})$ to $B_k$;
12   $\mathfrak{n} = p(\mathfrak{n})$;
until $p(\mathfrak{n}) = 0$ and $u(\mathfrak{b}) = \mathsf{true}$ for all $\mathfrak{b}$ incident with $\mathfrak{n}$ ;
```

Algebraic transformation of the circuit DAE

Together with (4.31), equation (4.32) can be used to replace (3.24a). Note that the number of equations in (4.32) may be greater than the number of equations in (3.24a), since we may find more fundamental CV loops in $\mathfrak{G}$ than fundamental V loops in $\mathfrak{G}_{V-C}$. However, (4.32) is more advantageous than (3.24a) with respect to the proposed index reduction method, since it only uses capacitive branch voltages as variables. This enables us to proceed analogously to the case of LI cutsets. By Assumption A2 we are able to write the charge conservation law (3.21d) as

$$\mathbf{q} = \mathbf{q}_C(\mathbf{v}_C, t) =: \begin{bmatrix} \widehat{\mathbf{q}}_C(\widehat{\mathbf{v}}_C) \\ \widetilde{\mathbf{q}}_C(\widetilde{\mathbf{v}}_C) \\ \bar{\mathbf{q}}_C(\bar{\mathbf{v}}_C) \end{bmatrix} =: \begin{bmatrix} \widehat{\mathbf{q}} \\ \widetilde{\mathbf{q}} \\ \bar{\mathbf{q}} \end{bmatrix} .$$

Algorithm 5: Computation of $\mathbf{M}_{CV}$

```
Data: Network graph $\mathfrak{G}$
begin
1   identify all blocks of $\mathfrak{G}_{CV}$ by Algorithm 4;
2   foreach block $\mathfrak{K}$ do
3       if $\mathfrak{K}$ contains at least one voltage source then
4           initialize $\mathfrak{T}$ with the voltage sources in $\mathfrak{K}$;
5           foreach capacitive branch $\mathfrak{c}$ in $\mathfrak{K}$ do
6               if $\mathfrak{c}$ closes a loop $\mathfrak{m}$ with $\mathfrak{T}$ then save all branches belonging to $\mathfrak{m}$;
7               else add $\mathfrak{c}$ to $\mathfrak{T}$;
8   define the coefficients of $\mathbf{M}_{CV}$ by (3.6);
end
```

Taking a look at (4.32), we see that we need the derivatives of the charge conservation law that are related to $\widehat{\mathbf{v}}_C$ and $\widetilde{\mathbf{v}}_C$. Thus, we get

$$\widehat{\mathbf{i}}_C = \frac{d}{dt}\widehat{\mathbf{q}} = \frac{\partial}{\partial \mathbf{v}_C}\widehat{\mathbf{q}}_C(\mathbf{v}_C)\frac{d}{dt}\widehat{\mathbf{v}}_C =: \widehat{\mathbf{C}}(\mathbf{v}_C)\frac{d}{dt}\widehat{\mathbf{v}}_C, \tag{4.33a}$$

$$\widetilde{\mathbf{i}}_C = \frac{d}{dt}\widetilde{\mathbf{q}} = \frac{\partial}{\partial \mathbf{v}_C}\widetilde{\mathbf{q}}_C(\mathbf{v}_C)\frac{d}{dt}\widetilde{\mathbf{v}}_C =: \widetilde{\mathbf{C}}(\mathbf{v}_C)\frac{d}{dt}\widetilde{\mathbf{v}}_C, \tag{4.33b}$$

where, by Assumption A1, the Jacobians $\widehat{\mathbf{C}}(\widehat{\mathbf{v}}_C)$ and $\widetilde{\mathbf{C}}(\widetilde{\mathbf{v}}_C)$ are positive definite. We use (4.33) and transform (4.32) as follows

$$\begin{aligned}
&\mathbf{0} = \frac{d}{dt}\widehat{\mathbf{v}}_C + \widetilde{\mathbf{M}}_C\frac{d}{dt}\widetilde{\mathbf{v}}_C + \mathbf{M}_{V,ind}\frac{d}{dt}\mathbf{v}_{ind}\\
\Longleftrightarrow &\mathbf{0} = \widehat{\mathbf{C}}(\widehat{\mathbf{v}}_C)\frac{d}{dt}\widehat{\mathbf{v}}_C + \widehat{\mathbf{C}}(\widehat{\mathbf{v}}_C)\widetilde{\mathbf{M}}_C\frac{d}{dt}\widetilde{\mathbf{v}}_C + \widehat{\mathbf{C}}(\widehat{\mathbf{v}}_C)\mathbf{M}_{V,ind}\frac{d}{dt}\mathbf{v}_{ind}(t)\\
\Longleftrightarrow &\mathbf{0} = \frac{d}{dt}\widehat{\mathbf{q}}_C + \underbrace{\widehat{\mathbf{C}}(\widehat{\mathbf{v}}_C)\widetilde{\mathbf{M}}_C\widetilde{\mathbf{C}}^{-1}(\widetilde{\mathbf{v}}_C)\widetilde{\mathbf{i}}_C + \widehat{\mathbf{C}}(\widehat{\mathbf{v}}_C)\mathbf{M}_{V,ind}\frac{d}{dt}\mathbf{v}_{ind}(t)}_{=:-\mathcal{I}_C(\widehat{\mathbf{v}}_C,\widetilde{\mathbf{v}}_C,\widetilde{\mathbf{i}}_C,t)}\,.
\end{aligned} \tag{4.34}$$

If we solve for $\frac{d}{dt}\widehat{\mathbf{q}}$ and use (4.31), then we finally obtain

$$\frac{d}{dt}\widehat{\mathbf{q}} = \mathcal{I}_C\left(\widehat{\mathbf{A}}_C^T\mathbf{e}, \widetilde{\mathbf{A}}_C^T\mathbf{e}, \widetilde{\mathbf{i}}_C, t\right). \tag{4.35}$$

This can be inserted into (3.21a) and we obtain

$$\begin{aligned}
\mathbf{0} &= \widetilde{\mathbf{A}}_C \frac{d}{dt}\widetilde{\mathbf{q}} + \bar{\mathbf{A}}_C \frac{d}{dt}\bar{\mathbf{q}} + \mathbf{A}_R \mathbf{g}(\mathbf{A}_R^T \mathbf{e}, t) + \mathbf{A}_L \mathbf{i}_L + \mathbf{A}_V \mathbf{i}_V \\
&\quad + \mathbf{A}_I \mathbf{i}_s(*, t) + \widehat{\mathbf{A}}_C \mathcal{I}_C \left(\widehat{\mathbf{A}}_C^T \mathbf{e}, \widetilde{\mathbf{A}}_C^T \mathbf{e}, \widetilde{\mathbf{i}}_C, t\right), && \text{(4.36a)} \\
\mathbf{0} &= \frac{d}{dt}\mathbf{\Phi} - \mathbf{A}_L^T \mathbf{e}, && \text{(4.36b)} \\
\mathbf{0} &= \mathbf{A}_V^T \mathbf{e} - \mathbf{v}_s(*, t), && \text{(4.36c)} \\
\mathbf{0} &= \widehat{\mathbf{q}} - \widehat{\mathbf{q}}_C \left(\widehat{\mathbf{A}}_C^T \mathbf{e}\right), && \text{(4.36d)} \\
\mathbf{0} &= \widetilde{\mathbf{q}} - \widetilde{\mathbf{q}}_C \left(\widetilde{\mathbf{A}}_C^T \mathbf{e}\right), && \text{(4.36e)} \\
\mathbf{0} &= \bar{\mathbf{q}} - \bar{\mathbf{q}}_C \left(\bar{\mathbf{A}}_C^T \mathbf{e}\right), && \text{(4.36f)} \\
\mathbf{0} &= \mathbf{\Phi} - \mathbf{\Phi}_L(\mathbf{i}_L). && \text{(4.36g)}
\end{aligned}$$

This is analogous to (4.14), however here we are faced with the problem that the capacitive current $\widetilde{\mathbf{i}}_C$ does not occur in the original DAE (3.21) as an unknown. We could use (4.33a) to replace $\widetilde{\mathbf{i}}_C$ by $\frac{d}{dt}\widetilde{\mathbf{q}}$, but that would introduce a derivative to (4.35) and lead to a time and state dependent mass matrix. Instead, we use a technique that is widely used in the design of circuits to measure the currents through those capacitances which are part of CV loops but do not close those loops. This technique uses the fact that the current through voltage sources is included in the MNA and MNA c/f as variable and that the currents through two elements in series are identical. Thus, to measure a current through a resistance or a capacitance, an additional voltage source is introduced in series to the respective element to the circuit. The voltage source adds a voltage of $\mathbf{v}_0 = 0$ to the circuit. Hence the new voltage source does not change the overall behavior of the circuit, but changes the analytical solution by introducing an additional variable $\mathbf{i}_{V_0} = \widetilde{\mathbf{i}}_C$. Figure 4.2 illustrates the changes to the circuit.

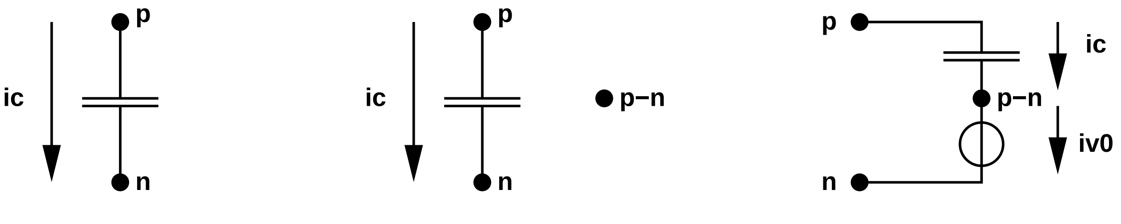


Figure 4.2.: Adding a voltage source to measure the current through a capacitance

Let $\widetilde{N}_C$ be the number of capacitances belonging to $\widetilde{\mathbf{A}}_C \in \mathbb{R}^{N \times \widetilde{N}_C}$. We will add a measuring voltage sources in series with each of the $\widetilde{N}_C$ capacitances that define $\widetilde{\mathbf{A}}_C$. To introduce these sources, we first have to introduce $\widetilde{N}_C$ new nodes. The node potentials of these new nodes will be denoted by $\mathbf{e}_0 \in \mathbb{R}^{\widetilde{N}_C}$ in the following and the new vector of all node potentials is given by

$$\mathbf{e}_{new} = \begin{bmatrix} \mathbf{e} \\ \mathbf{e}_0 \end{bmatrix} \in \mathbb{R}^{N+\widetilde{N}_C}. \tag{4.37a}$$

Since the new nodes are only incident with the newly introduced voltage sources and the capacitances that belong to $\widetilde{\mathbf{A}}_C$, the incidence matrices for the resistances and inductances changes as follows

$$\mathbf{A}_{R,new} = \begin{bmatrix} \mathbf{A}_R \\ \mathbf{0}_{\widetilde{N}_C \times N_R} \end{bmatrix}, \ \mathbf{A}_{L,new} = \begin{bmatrix} \mathbf{A}_L \\ \mathbf{0}_{\widetilde{N}_C \times N_L} \end{bmatrix}, \tag{4.37b}$$

where N_R is the number of resistances and N_L the number of inductances in the circuit. To define the new incidence matrices for capacitances and voltage sources, we first split $\widetilde{\mathbf{A}}_C$ into $\widetilde{\mathbf{A}}_{C+}$ and $\widetilde{\mathbf{A}}_{C-}$ such that

$$(\widetilde{a}_{C+})_{kl} = \begin{cases} 1, & \text{if } (\widetilde{a}_C)_{kl} = 1, \\ 0, & \text{else} \end{cases}$$

and

$$\widetilde{\mathbf{A}}_C = \widetilde{\mathbf{A}}_{C+} + \widetilde{\mathbf{A}}_{C-}.$$

This means that every column of $\widetilde{\mathbf{A}}_{C-}$ contains at most one non zero entry which is related to the node that the capacitance that belongs to that row enters. Analogously, every row of $\widetilde{\mathbf{A}}_{C+}$ contains at most one nonzero element related to the node the respective capacitance leaves. If a column in either $\widetilde{\mathbf{A}}_{C+}$ or $\widetilde{\mathbf{A}}_{C-}$ is zero, then the involved node is the reference node. Now, the new incidence matrices for the capacitances and the voltage sources can be written as

$$\mathbf{A}_{C,new} = \begin{bmatrix} \widetilde{\mathbf{A}}_{C+} & \bar{\mathbf{A}}_C \\ -\mathbf{I}_{\widetilde{N}_C} & \mathbf{0}_{\widetilde{N}_C \times \bar{N}_C} \end{bmatrix}, \ \mathbf{A}_{V,new} = \begin{bmatrix} \mathbf{A}_V & \widetilde{\mathbf{A}}_{C-} \\ \mathbf{0}_{\widetilde{N}_C \times N_V} & \mathbf{I}_{\widetilde{N}_C} \end{bmatrix}, \tag{4.37c}$$

where $\bar{N}_C$ is the number of capacitances that define $\bar{\mathbf{A}}_C$ and N_V the number of voltage sources in the original circuit. The currents through the voltage sources in the altered circuit are given by

$$\mathbf{i}_{V,new} = \begin{bmatrix} \mathbf{i}_V \\ \mathbf{i}_{V_0} \end{bmatrix} \tag{4.37d}$$

and the new function that describes the voltages of the voltage sources is given by

$$\mathbf{v}_{s,new}(*, t) = \begin{bmatrix} \mathbf{v}_s(*, t) \\ \mathbf{0} \end{bmatrix}.$$

In addition to this we set

$$\mathbf{q}_{new} = \begin{bmatrix} \widetilde{\mathbf{q}} \\ \bar{\mathbf{q}} \end{bmatrix}, \tag{4.37e}$$

$$\mathbf{q}_{C,new}\left(\begin{bmatrix} \widetilde{\mathbf{v}} \\ \bar{\mathbf{v}} \end{bmatrix}\right) = \begin{bmatrix} \widetilde{\mathbf{q}}_C(\widetilde{\mathbf{v}}) \\ \bar{\mathbf{q}}_C(\bar{\mathbf{v}}) \end{bmatrix}, \tag{4.37f}$$

$$\mathbf{C}_{new}\left(\begin{bmatrix} \widetilde{\mathbf{v}} \\ \bar{\mathbf{v}} \end{bmatrix}\right) = \begin{bmatrix} \widetilde{\mathbf{C}}(\widetilde{\mathbf{v}}) & \mathbf{0}_{\widetilde{N}_C \times \bar{N}_C} \\ \mathbf{0}_{\bar{N}_C \times \widetilde{N}_C} & \bar{\mathbf{C}}(\bar{\mathbf{v}}) \end{bmatrix}. \tag{4.37g}$$

Finally, we note that (4.35) suggests that the current $\widehat{\mathbf{i}}_C$ through the capacitances that close CV loops is determined by the voltage across those capacitances as well as by the currents and voltages of the remaining capacitances in the CV loops. Therefore, we are able to replace those capacitances by controlled sources that add the same current to the circuit and define

$$\mathbf{A}_{I,new} = \begin{bmatrix} \mathbf{A}_I & \widehat{\mathbf{A}}_C \\ \mathbf{0}_{\widetilde{N}_C \times N_I} & \mathbf{0}_{\widetilde{N}_C \times \widehat{N}_C} \end{bmatrix} \tag{4.37h}$$

and

$$\mathbf{i}_{s,new}(*,t) = \begin{bmatrix} \mathbf{i}_s(*,t) \\ \mathcal{I}_C\left(\widehat{\mathbf{A}}_C^T\mathbf{e}, \widetilde{\mathbf{A}}_C^T\mathbf{e}, \widetilde{\mathbf{i}}_C, t\right) \end{bmatrix} = \begin{bmatrix} \mathbf{i}_s(*,t) \\ \mathcal{I}_C\left(\widehat{\mathbf{A}}_C^T\mathbf{e}, \widetilde{\mathbf{A}}_C^T\mathbf{e}, \mathbf{i}_{V_0}, t\right) \end{bmatrix}. \tag{4.37i}$$

With the changes to the circuit that are described by (4.37) the DAE (4.36) becomes

$$\begin{aligned} \mathbf{0} &= \mathbf{A}_{C,new}\frac{d}{dt}\mathbf{q}_{new} + \mathbf{A}_{R,new}\mathbf{g}\left(\mathbf{A}_{R,new}^T\mathbf{e}_{new}, t\right) + \mathbf{A}_{L,new}\mathbf{i}_L \\ &\quad + \mathbf{A}_{V,new}\mathbf{i}_{V,new} + \mathbf{A}_{I,new}\mathbf{i}_{s,new}(*,t), \qquad (4.38a) \\ \mathbf{0} &= \frac{d}{dt}\mathbf{\Phi} - \mathbf{A}_{L,new}^T\mathbf{e}_{new}, \qquad (4.38b) \\ \mathbf{0} &= \mathbf{A}_{V,new}^T\mathbf{e}_{new} - \mathbf{v}_{s,new}(*,t), \qquad (4.38c) \\ \mathbf{0} &= \widehat{\mathbf{q}} - \widehat{\mathbf{q}}_C\left(\begin{bmatrix} \widehat{\mathbf{A}}_C^T & \mathbf{0}_{\widehat{N}_C \times \widetilde{N}_C} \end{bmatrix}\mathbf{e}_{new}\right), \qquad (4.38d) \\ \mathbf{0} &= \mathbf{q}_{new} - \mathbf{q}_{C,new}\left(\mathbf{A}_{C,new}^T\mathbf{e}_{new}\right), \qquad (4.38e) \\ \mathbf{0} &= \mathbf{\Phi} - \mathbf{\Phi}_L(\mathbf{i}_L). \qquad (4.38f) \end{aligned}$$

Remark 4.8. *System* (4.38) *contains* $2\widetilde{N}_C$ *additional equations compared with the original system. However, like the original system* (3.21)*, it has a constant mass matrix. Moreover, the additional equations for the new nodes and the new voltage sources are given by*

$$\mathbf{0} = -\frac{d}{dt}\widetilde{\mathbf{q}} + \mathbf{i}_{V_0} \text{ and } \mathbf{0} = \widetilde{\mathbf{A}}_{C-}^T\mathbf{e} + \mathbf{e}_0,$$

which are obviously both easy to solve.

Theorem 4.9. *Consider a circuit that fulfills both Assumption A1 (p. 58) and Assumption A2 (p. 73). Moreover, assume that the circuit contains CV loops, but no LI cutsets. If the functions of the sources are sufficiently smooth and* $\widetilde{\mathbf{C}}(\widetilde{\mathbf{v}}_C)$ *and* $\widehat{\mathbf{C}}(\widehat{\mathbf{v}}_C)$ *are differentiable, then the DAE* (4.38) *has differentiation index* 1.

Proof. Due to Table I.8 and Table I.9 in Appendix I, we can again assume that the original circuit does not contain any controlled sources. We start by considering the matrix

$$[\mathbf{A}_{C,new}\ \mathbf{A}_{R,new}\ \mathbf{A}_{V,new}] = \begin{bmatrix} \widetilde{\mathbf{A}}_{C+} & \bar{\mathbf{A}}_C & \mathbf{A}_R & \mathbf{A}_V & \widetilde{\mathbf{A}}_{C-} \\ -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{I} & -\widetilde{\mathbf{A}}_{C+} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \bar{\mathbf{A}}_C & \mathbf{A}_R & \mathbf{A}_V & \widetilde{\mathbf{A}}_C \\ -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}.$$

Since the branches that belong to the matrices $\widetilde{\mathbf{A}}_C$, $\bar{\mathbf{A}}_C$ and $\mathbf{A}_V$ form a tree $\mathfrak{T}_{CV}$ in $\mathfrak{G}_{CV}$, the matrix $\left[\widetilde{\mathbf{A}}_C\ \bar{\mathbf{A}}_C\ \mathbf{A}_V\right]$ has the same image as $[\mathbf{A}_C\ \mathbf{A}_V]$. Hence we have

$$\operatorname{range}\left[\widetilde{\mathbf{A}}_C\ \bar{\mathbf{A}}_C\ \mathbf{A}_R\ \mathbf{A}_V\right] = \operatorname{range}\left[\mathbf{A}_C\ \mathbf{A}_R\ \mathbf{A}_V\right].$$

Since we have assumed that the original circuit does not contain LI cutsets, $[\mathbf{A}_C\ \mathbf{A}_R\ \mathbf{A}_V]$ has full row rank and hence $\left[\widetilde{\mathbf{A}}_C\ \bar{\mathbf{A}}_C\ \mathbf{A}_R\ \mathbf{A}_V\right]$ has full row rank. This shows that $[\mathbf{A}_{C,new}\ \mathbf{A}_{R,new}\ \mathbf{A}_{V,new}]$ also has full row rank and, therefore, the altered circuit does not contain LI cutsets. Thus, $\mathbf{Z}_{CRV}$ vanishes. Moreover, $[\mathbf{A}_{C,new}\ \mathbf{A}_{V,new}] = \left[\widetilde{\mathbf{A}}_C\ \bar{\mathbf{A}}_C\ \mathbf{A}_V\right]$ has full column rank because there are no loops in $\mathfrak{T}_{CV}$ by definition. Therefore, there are no CV loops in the altered circuit and $\bar{\mathbf{Z}}_{V-C}$ also vanishes.

We apply Definition 3.41 to the altered circuit and define $\mathbf{W}$, $\mathbf{W}^{-1}\mathbf{e}$ and the matrices $\mathbf{W}_i,\ i = 1, 2, 3$ in analogy to (4.15) and (4.17). By multiplying (4.38a) with $\mathbf{W}^T$ and applying (4.17) to the whole system (4.38) we get

$$\mathbf{0} = \mathbf{W}_1^T \Big[\mathbf{A}_{C,new} \frac{d}{dt}\mathbf{q}_{new} + \mathbf{A}_{L,new}\mathbf{i}_L + \mathbf{A}_{V,new}\mathbf{i}_{V,new} + \mathbf{A}_{I,new}\mathbf{i}_{s,new}(t) + \mathbf{A}_{R,new}\mathbf{g}\left(\mathbf{A}_{R,new}^T\left(\mathbf{W}_1\mathbf{e}_C + \mathbf{W}_2\mathbf{e}_{V-C} + \mathbf{W}_3\mathbf{e}_{R-CV}\right), t\right)\Big], \tag{4.39a}$$

$$\mathbf{0} = \mathbf{W}_2^T \left[\mathbf{A}_{L,new}\mathbf{i}_L + \mathbf{A}_{V,new}\mathbf{i}_{V,new} + \mathbf{A}_{I,new}\mathbf{i}_{s,new}(t) + \mathbf{A}_{R,new}\mathbf{g}\left(\mathbf{A}_{R,new}^T\left(\mathbf{W}_1\mathbf{e}_C + \mathbf{W}_2\mathbf{e}_{V-C} + \mathbf{W}_3\mathbf{e}_{R-CV}\right), t\right)\right], \tag{4.39b}$$

$$\mathbf{0} = \mathbf{W}_3^T \left[\mathbf{A}_{L,new}\mathbf{i}_L + \mathbf{A}_{I,new}\mathbf{i}_{s,new}(t) + \mathbf{A}_{R,new}\mathbf{g}\left(\mathbf{A}_{R,new}^T\left(\mathbf{W}_1\mathbf{e}_C + \mathbf{W}_2\mathbf{e}_{V-C} + \mathbf{W}_3\mathbf{e}_{R-CV}\right), t\right)\right], \tag{4.39c}$$

and

$$\mathbf{0} = \frac{d}{dt}\mathbf{\Phi} - \mathbf{A}_{L,new}^T\left(\mathbf{W}_1\mathbf{e}_C + \mathbf{W}_2\mathbf{e}_{V-C} + \mathbf{W}_3\mathbf{e}_{R-CV}\right), \tag{4.39d}$$

$$\mathbf{0} = \mathbf{A}_{V,new}^T\left(\mathbf{W}_1\mathbf{e}_C + \mathbf{W}_2\mathbf{e}_{V-C}\right) - \mathbf{v}_{s,new}(t), \tag{4.39e}$$

$$\mathbf{0} = \widehat{\mathbf{q}} - \widehat{\mathbf{q}}_C\left(\left[\widehat{\mathbf{A}}_C^T \; \mathbf{0}_{\widehat{N}_C \times \widetilde{N}_C}\right]\left(\mathbf{W}_1\mathbf{e}_C + \mathbf{W}_2\mathbf{e}_{V-C} + \mathbf{W}_3\mathbf{e}_{R-CV}\right)\right), \tag{4.39f}$$

$$\mathbf{0} = \mathbf{q}_{new} - \mathbf{q}_{C,new}\left(\mathbf{A}_{C,new}^T\mathbf{W}_1\mathbf{e}_C\right), \tag{4.39g}$$

$$\mathbf{0} = \mathbf{\Phi} - \mathbf{\Phi}_L(\mathbf{i}_L). \tag{4.39h}$$

The derivative of (4.39h) together with (4.39d) yields

$$\mathbf{L}(\mathbf{i}_L)\frac{d}{dt}\mathbf{i}_L = \mathbf{A}_{L,new}^T\left(\mathbf{W}_1\mathbf{e}_c + \mathbf{W}_2\mathbf{e}_{V-C} + \mathbf{W}_{R-CV}\mathbf{e}_{R-CV}\right) \tag{4.40a}$$

and

$$\frac{d}{dt}\mathbf{\Phi} = \mathbf{A}_{L,new}^T\left(\mathbf{W}_1\mathbf{e}_c + \mathbf{W}_2\mathbf{e}_{V-C} + \mathbf{W}_{R-CV}\mathbf{e}_{R-CV}\right), \tag{4.40b}$$

which are expressions for $\frac{d}{dt}\mathbf{\Phi}$ and $\frac{d}{dt}\mathbf{i}_L$. Likewise, the derivative of (4.39g) together with (4.39a) leads to

$$\mathcal{C}\frac{d}{dt}\mathbf{e}_C = -\mathbf{W}_1^T\left[\mathbf{A}_{L,new}\mathbf{i}_L + \mathbf{A}_{V,new}\mathbf{i}_{V,new} + \mathbf{A}_{I,new}\mathbf{i}_{s,new}(t) + \mathbf{A}_{R,new}\mathbf{g}\left(\mathbf{A}_R^T\left(\mathbf{W}_1\mathbf{e}_C + \mathbf{W}_2\mathbf{e}_{V-C} + \mathbf{W}_3\mathbf{e}_{R-CV}\right), t\right)\right] \tag{4.40c}$$

with $\mathcal{C} := \mathbf{W}_1^T\mathbf{A}_{C,new}\mathbf{C}\left(\mathbf{A}_{C,new}^T\mathbf{W}_1^T\mathbf{e}_C\right)\mathbf{A}_{C,new}^T\mathbf{W}_1^T$ and

$$\frac{d}{dt}\mathbf{q}_{new} = -\mathbf{C}(\mathbf{A}_{C,new}^T\mathbf{W}_1\mathbf{e}_C)\mathbf{A}_{C,new}^T\mathbf{W}_1\mathcal{C}^{-1}\mathbf{W}_1^T \cdot \left[\mathbf{A}_{L,new}\mathbf{i}_L + \mathbf{A}_{V,new}\mathbf{i}_{V,new} + \mathbf{A}_{I,new}\mathbf{i}_{s,new}(t) + \mathbf{A}_{R,new}\mathbf{g}\left(\mathbf{A}_R^T\left(\mathbf{W}_1\mathbf{e}_C + \mathbf{W}_2\mathbf{e}_{V-C} + \mathbf{W}_3\mathbf{e}_{R-CV}\right), t\right)\right], \tag{4.40d}$$

since $\mathcal{C}$ is nonsingular by the construction of $\mathbf{W}_1 = \mathbf{W}_C$. Since, $\mathbf{A}_{V,new}^T \mathbf{W}_2$ is nonsingular by construction of $\mathbf{W}_{V-C}$, the derivative of (4.39e) yields

$$\begin{aligned}\mathbf{A}_{V,new}^T \mathbf{W}_2 \frac{d}{dt}\mathbf{e}_{V-C} = \mathbf{A}_{V,new}^T \mathbf{W}_1 \mathcal{C}^{-1} \mathbf{W}_1^T \big[&\mathbf{A}_{L,new}\mathbf{i}_L + \mathbf{A}_{V,new}\mathbf{i}_{V,new} + \mathbf{A}_{I,new}\mathbf{i}_{s,new}(t) \\ &+\mathbf{A}_{R,new}\mathbf{g}(\mathbf{A}_R^T\mathbf{e}, t)\big] + \frac{d}{dt}\mathbf{v}_{s,new}(t),\end{aligned} \tag{4.40e}$$

which is a differential equation for $\mathbf{e}_{V-C}$. Since the replaced capacitances were part of CV loops, we have

$$\left[\widehat{\mathbf{A}}_C^T \; \mathbf{0}_{\widehat{N}_C \times \widetilde{N}_C}\right] \mathbf{W}_3 = \mathbf{0},$$

and hence the derivative of (4.39f) is given by

$$\frac{d}{dt}\widehat{\mathbf{q}} = \widehat{\mathbf{C}}\left(\left[\widehat{\mathbf{A}}_C^T \; \mathbf{0}_{\widehat{N}_C \times \widetilde{N}_C}\right](\mathbf{W}_1\mathbf{e}_C + \mathbf{W}_2\mathbf{e}_{V-C})\right)\left[\mathbf{W}_1\frac{d}{dt}\mathbf{e}_C + \mathbf{W}_2\frac{d}{dt}\mathbf{e}_{V-C}\right]. \tag{4.40f}$$

Inserting (4.40b) and (4.40d) into (4.40f), we obtain a differential equation for $\frac{d}{dt}\widehat{\mathbf{q}}$. Now we differentiate (4.39c) and observe that $\mathcal{G} = \mathbf{W}_3^T \mathbf{A}_R \mathbf{G}(\mathbf{A}_R^T\mathbf{e}, t)\mathbf{W}_3$ is nonsingular by construction of $\mathbf{W}_{R-CV}$. In addition, we have

$$\begin{aligned}\mathbf{W}_3^T \mathbf{A}_{I,new}\frac{d}{dt}\mathbf{i}_{s,new}(t) &= \mathbf{W}_3^T \begin{bmatrix} \mathbf{A}_I & \widehat{\mathbf{A}}_C \\ \mathbf{0}_{\widetilde{N}_C \times N_I} & \mathbf{0}_{\widetilde{N}_C \times \widehat{N}_C}\end{bmatrix} \frac{d}{dt}\begin{bmatrix} \mathbf{i}_s(*, t) \\ \mathcal{I}_C\left(\widehat{\mathbf{A}}_C^T\mathbf{e}, \widetilde{\mathbf{A}}_C^T\mathbf{e}, \mathbf{i}_{V_0}, t\right)\end{bmatrix} \\ &= \mathbf{W}_3^T \begin{bmatrix} \mathbf{A}_I \\ \mathbf{0}_{\widetilde{N}_C \times N_I}\end{bmatrix}\frac{d}{dt}\mathbf{i}_s(t).\end{aligned}$$

Thus, the derivative of (4.39c) is given by

$$\begin{aligned}\mathcal{G}\frac{d}{dt}\mathbf{e}_{R-CV} = -\mathbf{W}_3^T \Bigg(&\mathbf{A}_{L,new}\frac{d}{dt}\mathbf{i}_L + \begin{bmatrix} \mathbf{A}_I \\ \mathbf{0}_{\widetilde{N}_C \times N_I}\end{bmatrix}\frac{d}{dt}\mathbf{i}_s(t) + \mathbf{A}_{R,new}^T \mathbf{g}_t'(\mathbf{A}_R^T\mathbf{e}, t) \\ &+ \mathbf{A}_{R,new}^T \mathbf{G}\left(\mathbf{A}_R^T\mathbf{e}, t\right)\mathbf{A}_{R,new}^T\left(\mathbf{W}_1\frac{d}{dt}\mathbf{e}_C + \mathbf{W}_2\frac{d}{dt}\mathbf{e}_{V-C}\right)\Bigg),\end{aligned} \tag{4.40g}$$

with $\mathbf{g}_t'(\mathbf{v}_R, t)$ as defined in (3.18). Using (4.40a), (4.40c) and (4.40d), equation (4.40g) leads to a differential equation for $\mathbf{e}_{R-CV}$. Moreover, with the derivative of (4.39f) we obtain a differential equations for $\frac{d}{dt}\widehat{\mathbf{q}}$. This is given by

$$\begin{aligned}\mathbf{0} = \mathbf{W}_2^T \Bigg[&\mathbf{A}_{V,new}\frac{d}{dt}\mathbf{i}_{V,new} + \mathbf{A}_{I,new}\frac{d}{dt}\mathbf{i}_{s,new}(t) + \mathbf{A}_{L,new}\frac{d}{dt}\mathbf{i}_L + \mathbf{A}_{R,new}\mathbf{g}_t'\left(\mathbf{A}_R^T\mathbf{e}, t\right) \\ &+ \mathbf{A}_{R,new}\mathbf{G}\left(\mathbf{A}_R^T e, t\right)\mathbf{A}_{R,new}^T\left(\mathbf{W}_1\frac{d}{dt}\mathbf{e}_C + \mathbf{W}_2\frac{d}{dt}\mathbf{e}_{V-C} + \mathbf{W}_3\frac{d}{dt}\mathbf{e}_{R-CV}\right)\Bigg].\end{aligned} \tag{4.40h}$$

To simplify the notation, in the following we set

$$\begin{aligned}
\mathbf{T}_1(\widehat{\mathbf{v}}_C,\widetilde{\mathbf{v}}_C) :=& \widehat{\mathbf{C}}(\widehat{\mathbf{v}}_C)\widetilde{\mathbf{M}}_C\widetilde{\mathbf{C}}^{-1}(\widetilde{\mathbf{v}}_C),\\
\mathbf{T}_2(\widehat{\mathbf{v}}_C) :=& \widehat{\mathbf{C}}(\widehat{\mathbf{v}}_C)\mathbf{M}_V,\\
\mathcal{T}_3 :=& \left[\frac{\partial}{\partial\widehat{\mathbf{v}}_C}\mathbf{T}_1(\widehat{\mathbf{A}}_C^T\mathbf{e},\widetilde{\mathbf{A}}_C^T\mathbf{e})\widehat{\mathbf{A}}_C^T\left(\mathbf{W}_1\frac{d}{dt}\mathbf{e}_C+\mathbf{W}_2\frac{d}{dt}\mathbf{e}_{V-C}\right)\right]\\
&+\left[\frac{\partial}{\partial\widetilde{\mathbf{v}}_C}\mathbf{T}_1(\widehat{\mathbf{A}}_C^T\mathbf{e},\widetilde{\mathbf{A}}_C^T\mathbf{e})\widetilde{\mathbf{A}}_C^T\mathbf{e}\mathbf{W}_1\frac{d}{dt}\mathbf{e}_C\right],\\
\mathcal{T}_4 :=& \left[\frac{\partial}{\partial\widehat{\mathbf{v}}_C}\mathbf{T}_2(\widehat{\mathbf{A}}_C^T\mathbf{e})\widehat{\mathbf{A}}_C^T\left(\mathbf{W}_1\frac{d}{dt}\mathbf{e}_C+\mathbf{W}_2\frac{d}{dt}\mathbf{e}_{V-C}\right)\right].
\end{aligned}$$

With this, we obtain

$$\begin{aligned}
\mathbf{W}_2^T&\left[\mathbf{A}_{V,new}\frac{d}{dt}\mathbf{i}_{V,new}+\mathbf{A}_{I,new}\frac{d}{dt}\mathbf{i}_{s,new}(t)\right]=\\
&=\mathbf{W}_2^T\left(\begin{bmatrix}\mathbf{A}_V & \widetilde{\mathbf{A}}_{C-}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}\frac{d}{dt}\begin{bmatrix}\mathbf{i}_V\\ \mathbf{i}_{V_0}\end{bmatrix}+\begin{bmatrix}\widehat{\mathbf{A}}_I\\ \mathbf{0}\end{bmatrix}\frac{d}{dt}\mathbf{i}_s(t)\right.\\
&\quad\left.-\begin{bmatrix}\widehat{\mathbf{A}}_C\\ \mathbf{0}\end{bmatrix}\left(\mathcal{T}_3\mathbf{i}_{V_0}+\mathbf{T}_1\frac{d}{dt}\mathbf{i}_{V_0}+\mathcal{T}_4\frac{d}{dt}\mathbf{v}_s(t)+\mathbf{T}_2\frac{d^2}{dt^2}\mathbf{v}_s(t)\right)\right)\\
&=\mathbf{W}_2^T\left(\begin{bmatrix}\mathbf{A}_V & \widetilde{\mathbf{A}}_{C-}-\widehat{\mathbf{A}}_C\mathbf{T}_1\\ \mathbf{0} & \mathbf{I}\end{bmatrix}\frac{d}{dt}\begin{bmatrix}\mathbf{i}_V\\ \mathbf{i}_{V_0}\end{bmatrix}\right.\\
&\quad\left.-\begin{bmatrix}\widehat{\mathbf{A}}_C\\ \mathbf{0}\end{bmatrix}\mathcal{T}_3\mathbf{i}_{V_0}+\begin{bmatrix}\mathbf{A}_I & \widehat{\mathbf{A}}_C\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\frac{d}{dt}\mathbf{i}_s(t)\\ \mathcal{T}_4\frac{d}{dt}\mathbf{v}_s(t)+\mathbf{T}_2\frac{d^2}{dt^2}\mathbf{v}_s(t)\end{bmatrix}\right).
\end{aligned}$$

Since $\mathbf{W}_{V-C}$ was chosen such that for $\mathbf{W}_2=\mathbf{Z}_C\mathbf{W}_{V-C}$

$$\mathbf{W}_2^T\begin{bmatrix}\mathbf{A}_V & \widetilde{\mathbf{A}}_{C-}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}$$

is nonsingular,

$$\mathbf{W}_2^T\begin{bmatrix}\mathbf{A}_V & \widetilde{\mathbf{A}}_{C-}-\widehat{\mathbf{A}}_C\mathbf{T}_1\\ \mathbf{0} & \mathbf{I}\end{bmatrix}$$

is also nonsingular. Therefore we obtain a differential equation for $\mathbf{i}_{V,new}$ from (4.40h). Since we only have to differentiate each equation in the DAE (4.38) at most once, the DAE has differentiation index 1. □

Proposition 4.10. *The DAE* (4.38) *has a properly stated leading term.*

Proof. In analogy to the proof of Proposition 4.6, we rewrite (4.38) as

$$\begin{aligned}
\mathbf{0} &= \mathbf{A}_{C,new}\frac{d}{dt}\mathbf{q}_{C,new}\left(\mathbf{A}_{C,new}^T\mathbf{e}_{new}\right) + \mathbf{A}_{R,new}\mathbf{g}\left(\mathbf{A}_{R,new}^T\mathbf{e}_{new}, t\right) + \mathbf{A}_{L,new}\mathbf{i}_L \\
&\quad + \mathbf{A}_{V,new}\mathbf{i}_{V,new} + \mathbf{A}_{I,new}\mathbf{i}_{s,new}(*, t), \\
\mathbf{0} &= \frac{d}{dt}\mathbf{\Phi}_L(\mathbf{i}_L) - \mathbf{A}_{L,new}^T\mathbf{e}_{new}, \\
\mathbf{0} &= \mathbf{A}_{V,new}^T\mathbf{e}_{new} - \mathbf{v}_{s,new}(*, t), \\
\mathbf{0} &= \widehat{\mathbf{q}} - \widehat{\mathbf{q}}_C\left(\left[\widehat{\mathbf{A}}_C^T \; \mathbf{0}_{\widehat{N}_C\times\widetilde{N}_C}\right]\mathbf{e}_{new}\right)
\end{aligned} \tag{4.41}$$

and choose

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{C,new} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_L \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \; \mathbf{R} = \begin{bmatrix} \mathbf{A}_{C,new}^- \mathbf{A}_{C,new} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_L \end{bmatrix} \text{ and } \mathbf{d}(t, \mathbf{x}) = \begin{bmatrix} \mathbf{q}_{C,new}\left(\mathbf{A}_{C,new}^T\mathbf{e}_{new}\right) \\ \mathbf{\Phi}_L(\mathbf{i}_L) \end{bmatrix},$$

where $\mathbf{A}_{C,new}^-$ is the Moore-Penrose inverse of $\mathbf{A}_{C,new}$ and $\mathbf{x} = \left[\mathbf{e}_{new}^T, \mathbf{i}_L^T, \mathbf{i}_{V,new}^T\right]^T$. □

4.4. Index reduction by element replacement

The index reduction methods for DAEs arising from the MNA which have been presented in Section 4.2 already hinted to the fact that we may be able to find a connection between the index reduction methods and the transformation of the circuit under consideration. The index reduction method for DAEs that arise from the MNA c/f (cf. Section 4.3) made this connection even more obvious. Indeed, it is possible to modify the model of a circuit that yields a DAE with differentiation index 2 in such a way that the altered circuit still has the same analytical solution but yields a DAE with differentiation index 1, independent of whether the DAE has been derived using the MNA or the MNA c/f [11,12]. This approach is advantageous as it only requires some additions to the element libraries of most commercial circuit simulation software, but no changes to the software itself.

In the following, we will consider circuits that fulfill both Assumption A1 (cf. page 58) and Assumption A2 (cf. page 73).

4.4.1. Modifications to the netlist due to CV loops

Consider a circuit that contains a single CV loop with capacitive branches $C_0, \ldots, C_k$ and branches of voltage source $V_1, \ldots, V_l$. We will chose the capacitance C_0 to be the capacitance that defines the loop (cf. Definition 3.31). The capacitance values are denoted by $c_0(v_{C_0}), \ldots, c_k(v_{C_k})$ and the functions that describe the voltage sources by $v_1(t), \ldots, v_l(t)$. The coefficients $\alpha_i,\ i = 1, \ldots, k$ and $\widetilde{\alpha}_j,\ j = 1, \ldots, l$ are given by the directions of the capacitances and voltage sources with respect to the loop.

Again, as in Section 4.3.2, we will add a voltage source in series with each of the capacitances $C_1, \dots, C_k$ to measure the current through the capacitances. The directions of these new elements with respect to the loop are the same as the directions of the capacitances they belong to. The currents of these measuring sources belonging to the capacitances $C_j,\ j = 1, \dots, k$ are denoted by $\widehat{i}_{V,j}$. Kirchhoff's Voltage Law applied to this loop yields

$$\mathcal{I} = i_{C_0} = -\sum_{j=1}^{k} \alpha_j \frac{c_0\left(\left[\mathbf{A}_C^T\mathbf{e}\right]_0\right)}{c_j\left(\left[\mathbf{A}_C^T\mathbf{e}\right]_j\right)} \widehat{i}_{V,j} - \sum_{j=1}^{l} \widetilde{\alpha}_j c_0\left(\left[\mathbf{A}_C^T\mathbf{e}\right]_0\right) \frac{d}{dt} v_j(t), \tag{4.42}$$

where $\left(\left[\mathbf{A}_C^T\mathbf{e}\right]_j\right)$ denotes the j-the entry of the vector $\mathbf{A}_C^T\mathbf{e}$ which belongs to the capacitance C_j, $j = 0, \dots, k$. Equation (4.42) defines the current through the capacitance C_0. Since this current is fixed by the CV loop, the capacitance C_0 does not contribute to the dynamic behavior of the circuit and thus can be replaced by a current source that introduces the current $\mathcal{I}$ into the circuit. This changes the netlist of the circuit under consideration. As a result, we obtain a netlist that produces a DAE with differentiation index 1 when modeled by either MNA or MNA c/f.

The replacement can be done in two ways. If the circuit simulation software that is used to generate the DAE from a circuit netlist is extendable to include current sources that are controlled by voltages and currents of various elements in the circuit, then the selected capacitances are replaced by just one current source each.

If this is not possible, then we can replace the selected capacitances by several standard current controlled and independent current sources, provided the capacitances are linear and constant. The current through the selected current source (4.42) then takes the form

$$\mathcal{I} = i_{C_0} = -\sum_{j=1}^{k} \alpha_j \frac{c_0}{c_j} \widehat{i}_{V,j} - \sum_{j=1}^{l} \widetilde{\alpha}_j c_0 \frac{d}{dt} v_j(t). \tag{4.43}$$

Consider first the terms in (4.43)

$$-\widetilde{\alpha}_j c_0 \frac{d}{dt} v_j(t), \quad j = 1, \dots, l. \tag{4.44}$$

Each of these terms can be realized by an independent current source with $\widetilde{i}_{s,j}(t) = -\widetilde{\alpha}_j c_0 \frac{d}{dt} v_j(t), \quad j = 1, \dots, l$. Similarly, we realize the terms

$$-\alpha_j \frac{c_0}{c_j} \widehat{i}_{V,j}, \quad j = 1, \dots, k. \tag{4.45}$$

with help of current controlled current sources with $i_{s,j}(i_{V,j}) = -\alpha_j \frac{c_0}{c_j} \widehat{i}_{V,j}$, $j = 1, \dots, k$. The controlling current is given by the current $i_{V,j}$ and the current gain is given by $-\alpha_j \frac{c_0}{c_j}$.

Since the currents of the current sources $i_{s,j}(t)$, $j = 1, \dots, l$ and $i_{s,j}(i_{V,j})$, $j = 1, \dots, k$ are summed up, we need to place them in parallel to obtain (4.43) on the circuit level. The current sources will have the same orientation as the replaced capacitance.

Example 4.11. *Consider the circuit in Figure 4.3. Due to the CV loop, that includes the capacitances C_1 and C_2 and the voltage source V, both MNA and MNA c/f would yield DAEs of differentiation index 2.*

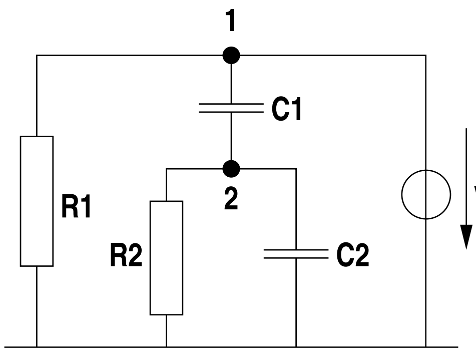


Figure 4.3.: Circuit with CV loop

If we assume that the capacitance C_2 defines the loop and that the orientations of the branches inside the loop are all positive, then we can replace C_2 by two current sources I_c and I_v. The currents of the current sources are defined by

$$i_v(t) = c_2 \frac{d}{dt} v(t)$$

for the current source I_v and

$$i_C(i_{V_0}) = -\frac{c_2}{c_1} i_{V_0}$$

for the current source I_C where i_{V_0} is the current through the newly introduced voltage source V_0 in series to the capacitance C_1. Figure 4.4 shows the circuit after the replacement. If the new circuit is modeled with either MNA or MNA c/f, then the resulting DAEs are of differentiation index 1.

Example 4.12. *The example of the NAND gate shown in Figure 4.6 is taken from [1]. Figure 4.5 shows the equivalent circuit for a MOSFET used in this example. The complete circuit is shown in Figure 4.6. Due to the CV loops which are contained in the circuit, the DAEs that arise from either MNA or MNA c/f have differentiation index 2.*

Algorithm 5 identifies six CV loops in the given circuit. The exact composition of these loops depends on the order in which the capacitances are examined. Here, we chose the CV loops that are listed in Table 4.1. The signs show the orientation of the elements inside the CV loops with respect to the selected capacitance which defines the CV loop.

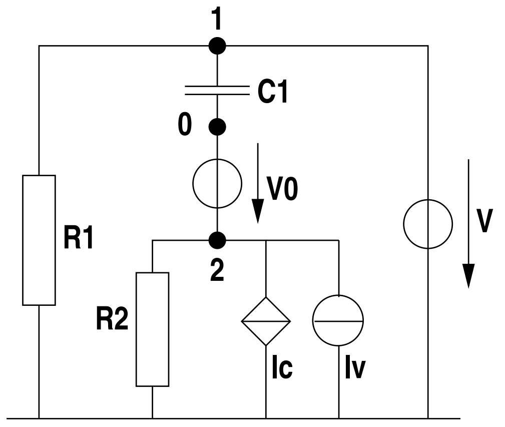


Figure 4.4.: Index reduced circuit

selected capacitance	remaining elements
C_{db_1}	$+C, +C_{gd_1}, -V_{bb}$
C_{sb_1}	$+C, +C_{gs_1}, -V_{bb}$
C_{db_2}	$+C_{gd_2}, +V_1, -V_{bb}$
C_{sb_2}	$+C_{gs_2}, +V_1, -V_{bb}$
C_{db_3}	$+C_{gd_3}, +V_2, -V_{bb}$
C_{sb_3}	$+C_{gs_3}, +V_2, -V_{bb}$

Table 4.1.: CV loops in the NAND gate

In order to reduce the index, the selected capacitances will be replaced by the controlled current sources that are listed in Table 4.2. Figure 4.7 shows the circuit after the replacement.

Remark 4.13. *Even though Example 4.12 suggests that there is a way to modify the individual MOSFETs regardless of the surrounding network in order to lower the index, this is not true. Table 4.2 shows that the functions for the current controlled sources not only depend on local elements inside a MOSFET, but also on elements of the surrounding network, like the input sources V_1 and V_2.*

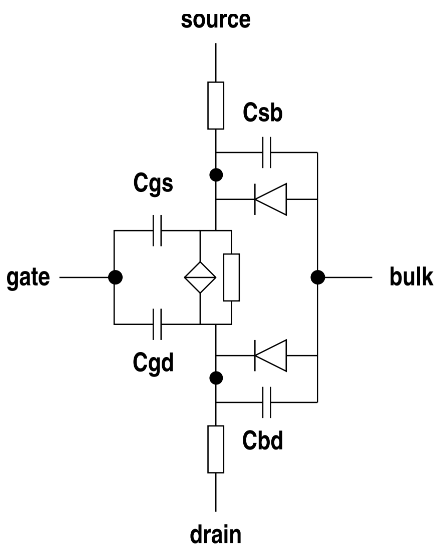


Figure 4.5.: Level B equivalent circuit for a MOSFET

selected capacitance	replacement
C_{db_1}	$\mathcal{I}_1 = i_{C_{db_1}} = -c_{db_1}\left(c^{-1} i_{V,C} + c_{gd_1}^{-1} i_{V,gd_1} - \frac{d}{dt} v_{bb}\right)$
C_{sb_1}	$\mathcal{I}_2 = i_{C_{sb_1}} = -c_{sb_1}\left(c^{-1} i_{V,C} + c_{gs_1}^{-1} i_{V,gs_1} - \frac{d}{dt} v_{bb}\right)$
C_{db_2}	$\mathcal{I}_3 = i_{C_{db_2}} = -c_{db_2}\left(c_{gd_2}^{-1} i_{V,gd_2} + \frac{d}{dt} v_1 - \frac{d}{dt} v_{bb}\right)$
C_{sb_2}	$\mathcal{I}_4 = i_{C_{sb_2}} = -c_{sb_2}\left(c_{gs_2}^{-1} i_{V,gs_2} + \frac{d}{dt} v_1 - \frac{d}{dt} v_{bb}\right)$
C_{db_3}	$\mathcal{I}_5 = i_{C_{db_3}} = -c_{db_3}\left(c_{gd_3}^{-1} i_{V,gd_3} + \frac{d}{dt} v_2 - \frac{d}{dt} v_{bb}\right)$
C_{sb_3}	$\mathcal{I}_6 = i_{C_{sb_3}} = -c_{sb_3}\left(c_{gs_3}^{-1} i_{V,gs_3} + \frac{d}{dt} v_2 - \frac{d}{dt} v_{bb}\right)$

Table 4.2.: Current controlled current sources to replace the selected capacitances

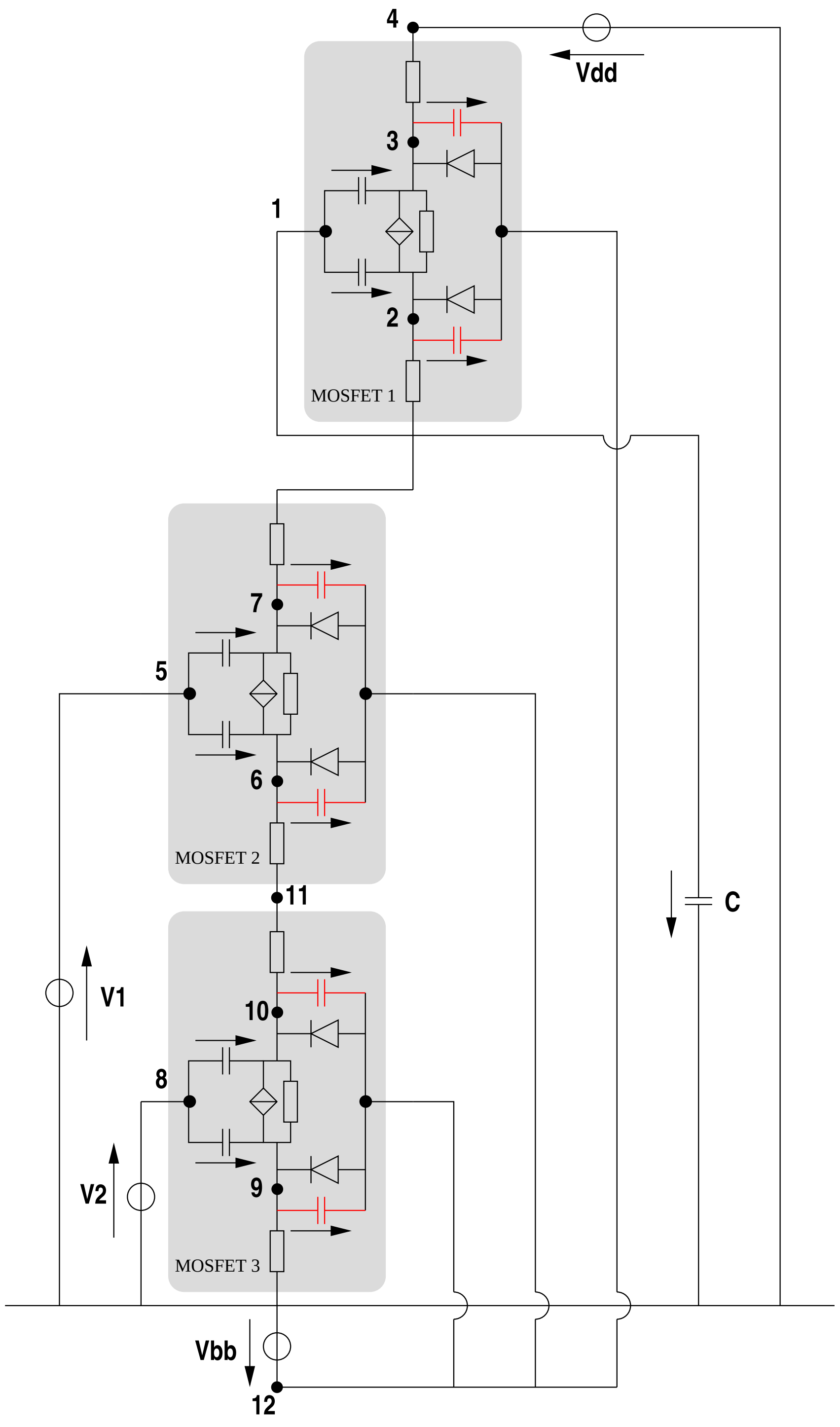


Figure 4.6.: NAND gate, differentiation index 2

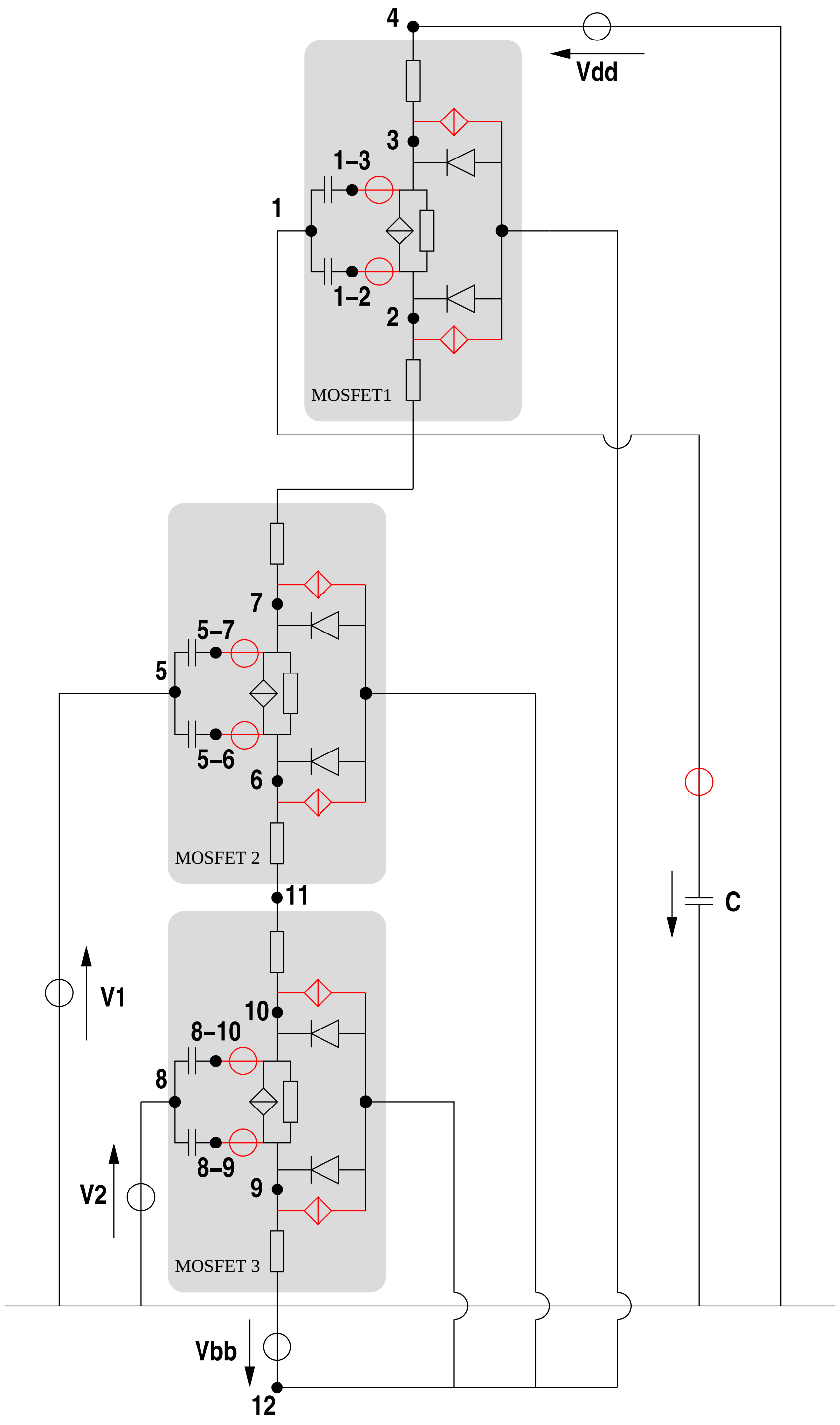


Figure 4.7.: NAND gate, differentiation index 1

4.4.2. Modifications to the netlist due to LI cutsets

The necessary modifications of the netlist in case of LI cutsets are similar to those that are necessary in case of CV loops. Again, we will assume that the circuit under consideration only contains one LI-cutset with inductances $L_0, \dots, L_k$ with inductance values $l_0(i_{L_0}), \dots, l_k(i_{L_k})$ and current sources $I_1, \dots, I_m$ with source functions $i_1(t), \dots, i_m(t)$. The inductance L_0 is taken as the inductance that defines the cutset. The directions of the inductances and current sources with respect to the cutset are denoted by α_j, $j = 1, \dots, k$ and $\widetilde{\alpha}_j$, $j = 1, \dots, m$, respectively. The voltage that arises from the LI cutset is given by

$$\mathcal{V} = v_{L_0} = -\sum_{j=1}^{k} \alpha_j \frac{l_0(i_{L_0})}{l_j(i_{L_j})} v_{L_j} - \sum_{j=1}^{m} \widetilde{\alpha}_j l_0(i_{L_0}) \frac{d}{dt} i_j(t). \tag{4.46}$$

Here, v_{L_j} denotes the voltage across the inductance L_j, $j = 1, \dots, k$. Equation (4.46) now defines the voltage across the inductance L_0 and hence, L_0 does not contribute to the dynamic behavior of the circuit. Thus, we are able to replace L_0 by a voltage source that introduces the voltage given by (4.46) into the circuit.

Again, the replacement can be either done directly if the software that is used to simulate the circuit is extendable. If this is not the case and the inductances in the cutset are linear, then we are again able to realize (4.46) on the netlist level by using several voltage controlled and independent sources. In this case, voltage controlled voltage sources $v_{s,j}(v_{L_j})$ with controlling voltages v_{L_j} and voltage gains $-\alpha_j \frac{l_0}{l_j}$ are used to express the terms

$$-\alpha_j \frac{l_0}{l_j} v_{L_j}, \quad j = 1, \dots, k.$$

For the terms

$$-\widetilde{\alpha}_j l_0 \frac{d}{dt} i_j(t), \quad j = 1, \dots, m$$

independent sources which are described by the functions $v_{s,j}(t)$ are used. Since the voltages of the voltage sources are summed up, the sources need to be placed in series and have the same orientations as the replaced inductance.

Example 4.14. *Consider the circuit in Figure 4.8. Due to the LI cutset which consists of the inductances L_1 and L_2 and the current source I_1 the circuit has differentiation index 2.*

If we assume that L_1 defines the cutset and that the orientations of the branches inside the cutset are all positive, then we can replace L_1 by two voltage sources V_L and V_I. The voltages of the voltage sources are defined by

$$v_I(t) = l_1 \frac{d}{dt} i_1(t)$$

for the voltage source V_I and

$$v_L(v_{L_2}) = -\frac{l_1}{l_2} v_{L_2} = -\frac{l_1}{l_2}(e_2 - e_3)$$

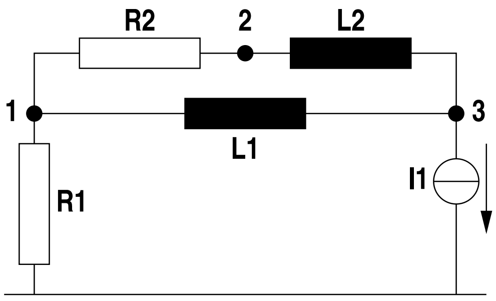


Figure 4.8.: Circuit with LI cutset

for the voltage source V_L. Figure 4.9 shows the circuit after the replacement. Again, both MNA and MNA c/f yield DAEs of differentiation index 1*.*

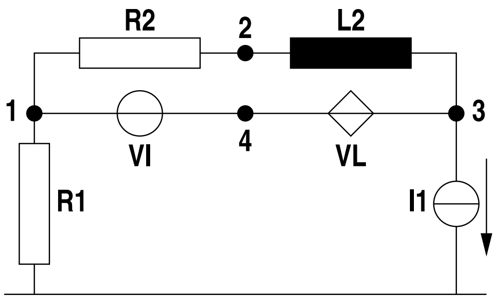


Figure 4.9.: Index reduced circuit

Example 4.15. *The oscillator in Figure 4.10 is taken from [24]. The circuit has differentiation index 2 due to the LI-cutsets that it contains. Table 4.3 displays one possible choice of the fundamental LI-cutsets.*

selected capacitance	remaining elements
L_d	$L_{g_1}, L_{g_2}, L_{s_1}$
L_s	L_{s_1}
L_g	L_{g_1}, L_{g_2}

Table 4.3.: LI cutsets in the oscillator circuit

In order to reduce the index, the selected inductances are replaced by voltage controlled voltage sources for which the describing functions are given in Table 4.4. Figure 4.11 shows the circuit after the replacement of the inductances.

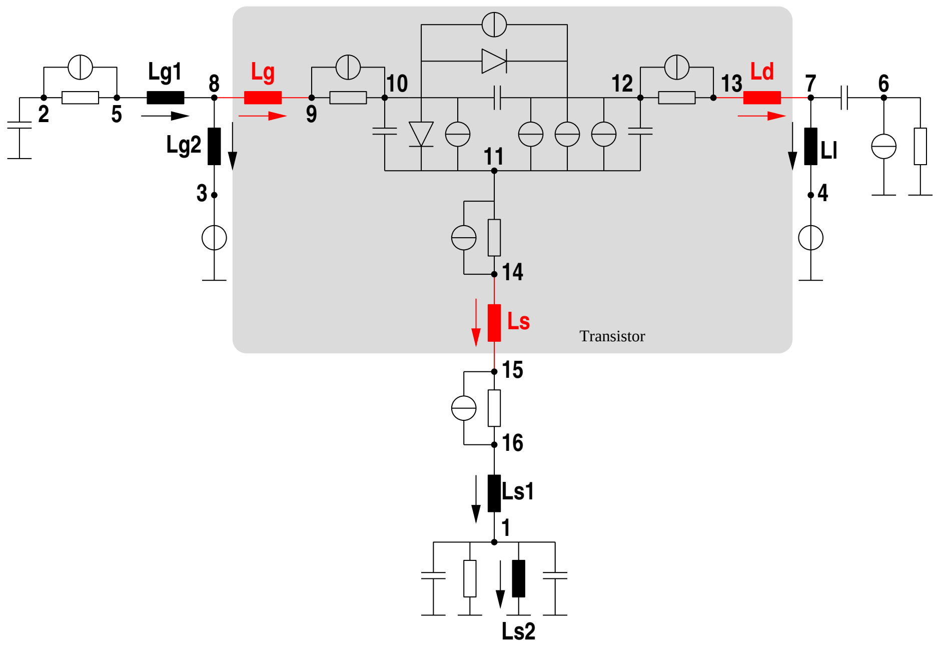


Figure 4.10.: Oscillator, differentiation index 2

selected capacitance	replacement
L_d	$\mathcal{V}_1 = v_{L_d} = -l_d\left(l_{g1}^{-1}v_{L_{g_1}} + l_{g2}^{-1}v_{L_{g_2}} + l_{s1}^{-1}v_{L_{s_1}}\right)$
L_s	$\mathcal{V}_2 = v_{L_s} = -l_d\left(l_{s1}^{-1}v_{L_{s_1}}\right)$
L_g	$\mathcal{V}_3 = v_{L_g} = -l_d\left(l_{g1}^{-1}v_{L_{g_1}} + l_{g2}^{-1}v_{L_{g_2}}\right)$

Table 4.4.: Voltage controlled voltage sources to replace the selected inductances

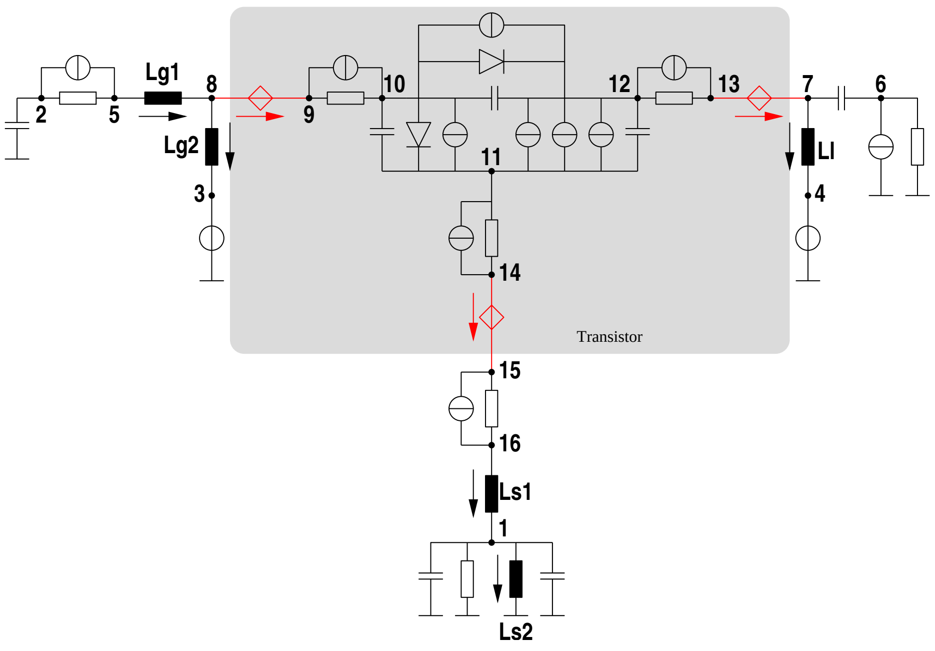


Figure 4.11.: Oscillator, differentiation index 1

5. qPsim - Software for circuit simulation

In this chapter, the academical circuit simulator qPsim will be presented. qPsim is based on the code Psim (Pedagogical SIMulator) [65] which has been written in C++ [74]. Psim features a built-in automatic differentiation to evaluate the involved functions as well as to compute their derivatives. qPsim builds upon this feature to compute the derivatives necessary for the index reduction.

The original code Psim created the circuit DAEs based on the Modified Nodal Analysis. qPsim has been changed in such a way that the circuit DAEs are created based on the charge-oriented Modified Nodal Analysis. In order to determine the differentiation index of a given circuit, Algorithms 3, 4 and 5 have been implemented in qPsim. To reduce the differentiation index if needed, the index reduction as proposed in Section 4.3 has been implemented.

This chapter is split into two parts. The first part consists of Section 5.1 and gives a detailed description of the qPsim code. Section 5.2 then discusses the performance of the new index reduction method implemented in qPsim by means of several examples.

5.1. Description of qPsim

5.1.1. Overview over the main classes in qPsim

qPsim is written in C++ and makes use of the programming language's object oriented features like class hierarchies and template classes. In this section, the most important classes in qPsim are described. These classes are defined in the following header files.

- `ad_double.h`
- `List.h`
- `hierarchy.h`
- `device.h`
- `topology.h`
- `minext.h`

ad_double.h

In `ad_double.h`, the classes for the automatic differentiation are defined. These classes include the class for the new datatype `ad_double` which models a function $\mathbf{f}(\mathbf{x})$ evaluated for a certain value $\mathbf{x}_0$ as well as the gradient of the function $\frac{\partial}{\partial \mathbf{x}}\mathbf{f}$ which also is evaluated at $\mathbf{x}_0$. This class also overrides the standard operators '+', '-', '·' and so on. Other classes defined in the same header file are designed to handle the information in the gradient (cf. Section 5.1.3). The interdependencies of the classes in `ad_double.h` are shown in Figure 5.1.

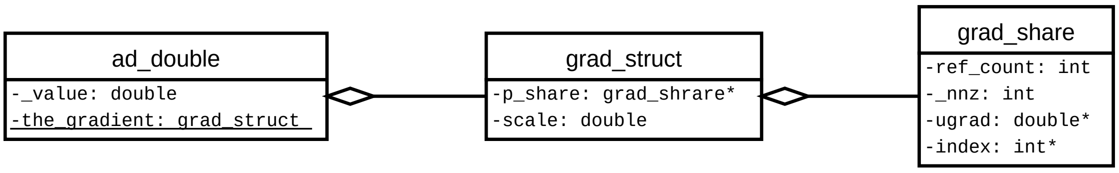


Figure 5.1.: Class hierarchy in `ad_double.h`

List.h

The classes defined in `List.h` are templates for lists of objects. These lists are used by the classes `Circuit` and `SubCircuit` defined in `hierarchy.h` to manage the elements and nodes in a circuit.

hierarchy.h

The classes which are defined in `hierarchy.h` are the central classes of QPSIM . These classes model the parts of a circuit as well as the circuit itself. All classes are derived from the root class `HierObj`. This root class is implemented in such a way that a given circuit is represented as a rooted tree of objects with the nodes and elements of the circuit as leaves. The class `residual` plays a special role. It is used in the class `Circuit` to manage the equations that are associated with the circuit. Since the circuit is modeled by MNA c/f, these equations take the form

$$\mathbf{Q}\frac{d}{dt}\mathbf{x} + \mathbf{f}(t, \mathbf{x}) = 0, \tag{5.1}$$

where $\mathbf{Q}$ is a square, constant matrix that may be singular. The class `residual` has two members of type `ad_double`. The first `ad_double`, `f_part`, represents the part of an equation that does not depend on the derivatives of variables. The parts of an equation that depend on the derivatives of variables are gathered in the second member, `q_part`.

The class `Circuit` is the class that actually is designed to model a circuit. Because of its importance, we will list some of its methods that are used by the numerical methods.

int num_unk() returns the number of unknowns of the circuit DAE.

int num_rsd() returns the number of equations of the circuit DAE.

void show_unk(FILE*) writes the names, values and derivatives of the unknowns to `FILE`.

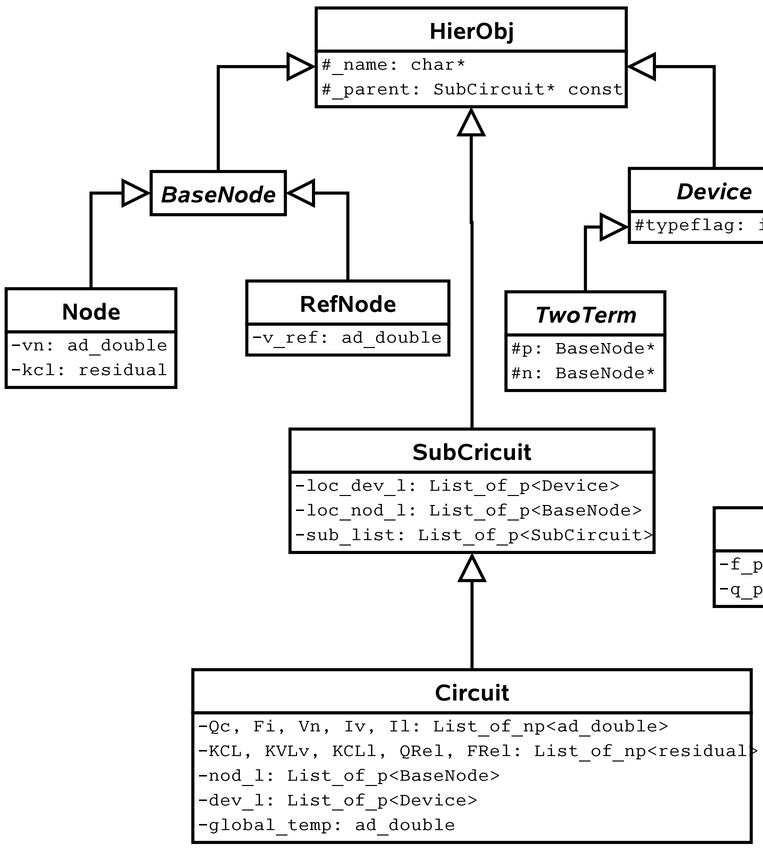


Figure 5.2.: Class hierarchy in `hierarchy.h`

void show_rsd(FILE*) writes the names, values and derivatives of the circuit DAE equations to `FILE`.

void vec_to_unk(double* Y) sets the values of the unknowns to `Y`.

void rsd_to_vec(eq_part CHOICE, double* Y) sets `Y` to $\mathbf{Q}\frac{d}{dt}\mathbf{x}$ if `CHOICE = Q_PART` and to $\mathbf{f}(t,\mathbf{x})$ if `CHOICE = F_PART`.

void D_dense(eq_part CHOICE , int LDA , int NCOL , double* JAC) sets `JAC` to Q if `CHOICE = Q_PART` and to $\mathbf{F}(t,\mathbf{x}) = \frac{\partial}{\partial x}\mathbf{f}(t,\mathbf{x})$ if `CHOICE = F_PART`. The Jacobians are stored columnwise.

void eval() evaluates the equations associated with the circuit and computes their derivatives at the same time.

device.h

This header file contains the definitions of the classes that model circuit elements like resistances, capacitances or sources. All classes are derived either from one of the classes `Device`, `TwoTerminal` and `TwoPort`, which are defined in `hierarchy.h`. In Appendix II, the most important elements are listed together with the constructors of their respective classes.

topology.h

This header file contains the classes and routines that perform the graph theoretical analysis of a given circuit. The methods used in the analysis are based on Algorithms 3, 4 and 5 (cf. Section 5.1.4).

minext.h

The functions in this header file are used to handle the equations that are necessary to perform the index reduction proposed in Chapter 4. The functions include `getCVeq` and `getLIeq`, which evaluate the additional equations, and the functions `hcMass`, `hcliFcn` and `hccvFcn`, which return the Jacobians and the values of the additional equations.

5.1.2. Netlist description of a circuit

To simulate a circuit with help of QPSIM , the circuit must be provided as a SPICE-like netlist (cf. [42]). To do so, the user has to define a new class which is derived from the `Circuit` class and thus inherits all of the methods that are defined by the `Circuit` class. If the circuit contains time-dependent sources, the class that models the circuit must implement the method `void time_stimulus(ad_double t)` to provide the values of sources at a given time t.

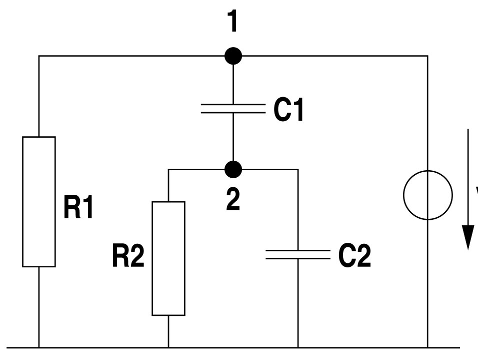


Figure 5.3.: Example circuit

Example 5.1. *Consider the circuit shown in Figure 5.3. This circuit is modeled by the following* QPSIM *class.*

```
class CVex : public Circuit
{
 public:
  // Reference node of the circuit with fixed node potential equal to 0.0
  RefNode gnd;
  // the remaining nodes of the circuit
  Node n1, n2;
  // voltage sources
  Vsrc v1;
  // linear resistances
  Res r1, r2;
  // linear capacitances
  Cap c1, c2;
  // the construcutor of the "CVex" class
  // the members of the class have to appear in the initializer list
  // in correct order
  CVex() : Circuit("Index2"),
     gnd(this, "gnd", 0.0),
     n1(this, "n1"),
     n2(this, "n2"),
     v1(this, "v1", n1, gnd, 1.0),
     r1(this, "r1", n1, gnd, 1.0),
     r2(this, "r2", n2, gnd, 1.0),
     c1(this, "c1", n1, n2, 1.0),
     c2(this, "c2", n2, gnd, 1.0)
  {}

  // function for the time-dependent voltage source
  void time_stimulus(ad_double t){v1.set_v(-sin(100.0*t));}
};
```

5.1.3. Creating equations from the netlist

The DAE that is used to simulate the circuit in qPsim arises from MNA c/f. It is evaluated whenever the `Circuit` object calls the method `eval()`. The evaluation is done on an element-by-element basis. This element-by-element approach is also called *stamping* of the circuit equations. It is a standard technique used in software for circuit simulation.

Data structure for the MNA c/f equations and unknown

In qPsim, the class `Node` has a member `kcl` of type `residual` that models the sum of currents that enter a node. When an object of type `Node` is created, a pointer to it will be placed into the list of nodes `nod_l` of the `Circuit` object the `Node` belongs to. Moreover, the `residual` object for the `Node` is also created and a pointer to that `residual` object is placed into the list of KCL equations `KCL` of the `Circuit`. Figure 5.4 shows the pointer lists and the pointers to the `Node` objects.

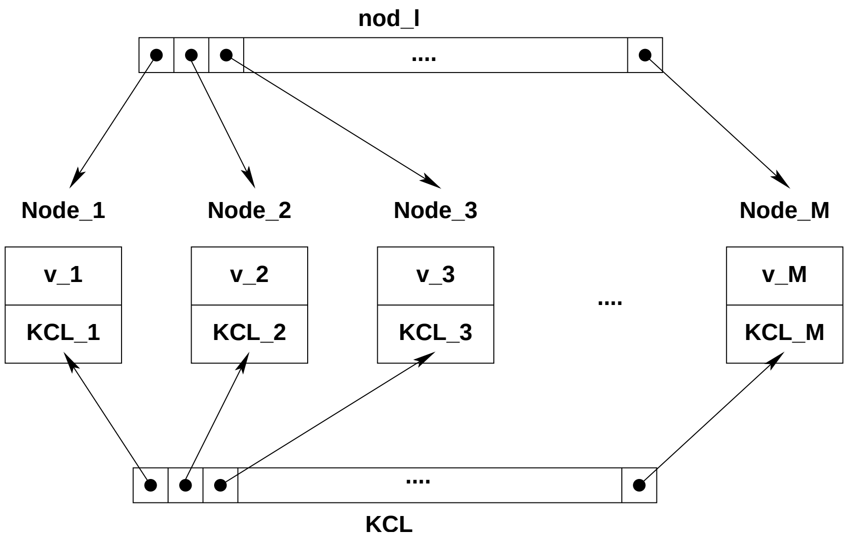


Figure 5.4.: Pointer lists to nodes and KCL equations

In addition to the `residual kcl` the class `Node` also has a member `v` of type `ad_double` which models the node potential of a specific node. When a `Node` object is created, the `ad_double` object will also be created and a pointer to this object will be placed in the list of node potentials `Vn` of the `Circuit` object (cf. Figure 5.2). The data structure that is build in this way is similar to the one for the `KCL residual` objects in Figure 5.4.

Classes that model electric elements have pointers to the incident nodes. If an explicit current-voltage relation exists for an electric element, then the class that models that element does neither have members of type `residual` that account for equations associated with that element nor members of type `ad_double` that account for additional unknowns. Such elements include for example resistances. In this case, the voltage across the element is computed from the node potentials of the incident nodes. This voltage then is used to compute the current through the element which is then added to the `residual kcl` of the incident nodes.

If on the other hand, the electric element does not have an explicit current-voltage relation, then the class that models such an element has both a member of type `residual` and a member of type `ad_double`. Consider for example the class `Vsrc` which models an independent voltage source. The class has a member `ad_double ib` and a member `residual kvl`. The member `ib` models the unknown for the current through the voltage source, whereas `kvl` models the equation associated with the voltage source. When an object of type `Vsrc` is created, the members are created as well. Moreover, pointers are placed into the list of voltage source currents `Iv` and the list of voltage source equations `KVLv` of the superior `Circuit` object. Similar, for the classes `Cap` and `Ind` which model capacitances and inductances, there are class members that model the charges and fluxes and the conservation laws that describe the elements. Figure 5.5 shows the data structure for one object of type `Vsrc`.

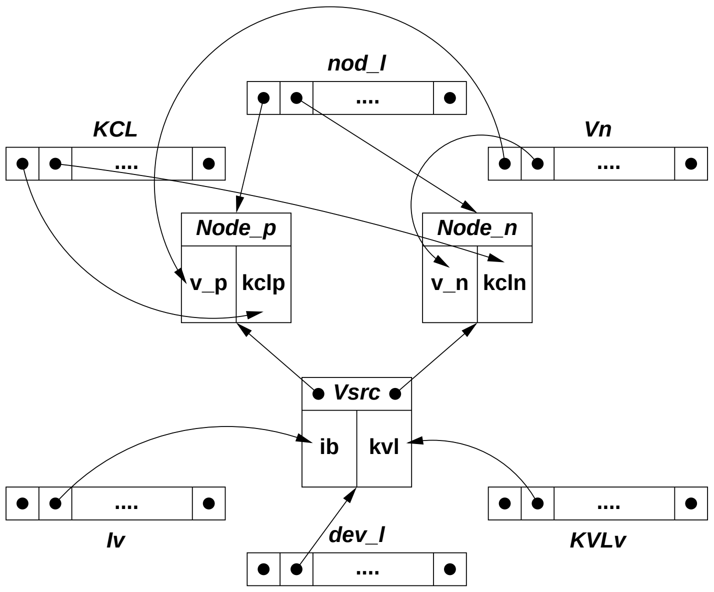


Figure 5.5.: Pointer lists for a `Vsrc` object

Automatic differentiation

If we want to simulate the transient behavior of a circuit, then we need the derivatives of certain variables with respect to the time t as well as the derivatives of the equations with respect to the variables or their derivatives. Since qPsim generates these equations and variables from a netlist, we cannot supply subroutines for these derivatives. Instead, qPsim uses automatic differentiation techniques to compute the needed derivatives. The automatic differentiation in qPsim is realized by defining the class `ad_double`. The operator overloading feature of C++ is used to implement the differentiation rules.

The class `ad_double` has two members. The first member, `_value`, is of type `double` and models a function value $\mathbf{f}(\mathbf{x}_0)$. The second member, `the_gradient`, is of type `grad_struct` and models the function's gradient $\frac{\partial}{\partial \mathbf{x}}\mathbf{f}(\mathbf{x}_0)$ (cf. Figure 5.1). The class defines operators to add, subtract, multiply and divide `ad_double` objects. These operators perform the normal operation on the function value and change the gradient accordingly. Let `X`, `Y` and `Z` be `ad_doubles`. Then the operations are defined as follows.

```
Z = X + Y            Z._value = X._value + Y._value
              Z.the_gradient = 1.0*X.the_gradient
                             + 1.0*Y.the_gradient
```

```
Z = X - Y             Z._value = X._value - Y._value
               Z.the_gradient = 1.0*X.the_gradient
                              - 1.0*Y.the_gradient


Z = X * Y             Z._value = X._value * Y._value
               Z.the_gradient = Y._value*X.the_gradient
                              + X._value*Y.the_gradient


Z = X / Y             Z._value = X._value / Y._value
               Z.the_gradient = (1.0/Y._value)*X.the_gradient
                              - (X._value/(Y._value*Y._value))
                                 *Y.the_gradient


Z = g(X)             Z._value = g(X._value)
              Z.the_gradient = g'(X._value)*X.the_gradient
```

Here g is one of the functions cos, sin, tan, atan, exp, log and pow. The last operation implements the differentiation by the chain rule. Note that in this case, Z.the_gradient is a multiple of X.the_gradient. The automatic differentiation in qPSIM makes use of this fact to save memory and computation time. To do so, the class grad_share is implemented to manage the unscaled values of the gradient and a reference counter. The class grad_struct has a pointer to grad_share and a double value to scale the gradient. This approach saves memory space and computation time, especially if we consider functions in n variables. In this case, the member the_gradient would be a double array of size n. Now, consider Z = g(X). If both Z and X had their own array the_gradient, then the array X.the_gradient would have to be copied to Z.the_gradient which has then to be multiplied with the factor g'(X._value). This is not only time consuming, but also results in the double amount of memory required to store both X.the_gradient and Z.the_gradient. To save the time and memory, both X.the_gradient and Z.the_gradient point to the same array. However, then the scaling of Z.the_gradient cannot be done directly. In qPSIM , the scaling factor is stored along with the pointer to the gradient array in a grad_struct for each object of type ad_double. The actual array is stored in grad_share. Also, each object of type grad_share has a counter which is incremented by one if new pointer to the object is added. As soon as an object that points to a grad_share is destroyed, the reference counter is decremented by one. The grad_share is destroyed if the reference counter reaches zero. In this way, only one array of size n has to be stored for both X.the_gradient and Z.the_gradient. If for example Z.the_gradient is requested by a function in qPSIM , then the array is multiplied by the scaling factor g'(X._value) and returned to the calling function. If Z.the_gradient is destroyed, then the reference counter for the array is lowered by one. However, since X.the_gradient is still pointing to the array, the array is not destroyed. If X.the_gradient is destroyed and no other object is pointing to the array, then the array is also destroyed.

5.1.4. Transient analysis and index reduction

Up to now, we have described how QPSIM creates the data structure that models the unknowns and equations of a circuit which has been modeled by MNA c/f. The actual simulation of the transient behavior of the circuit can now be done by calling the function `int transient(Circuit& W, double* x, PARAM p)`. The `PARAM` object contains the parameters that are passed to `transient`. They are listed in Table 5.1.

Parameter	**Feature**
`iout`	output 0: no output (default) 1: output only at computed steps 2: output at user provided points 3: output at both user provided and computed steps
`ir_flag`	index reduction 0: perform index reduction (default) 1: do not perform index reduction
`method`	discretization method used 0: RADAU5 (default) 1: DASPK3.1
`con_inv`	initial values 0: initial values are not consistent (default) 1: initial values are consistent
`grid_size`	number of user provided points for output
`t_start`	starting point for the transient analysis
`t_end`	end point for the transient analysis
`atol / rtol`	absolute and relative tolerances
`h0`	initial stepsize
`grid`	user provided points for output
`fname`	name of the output file(s)

Table 5.1.: Parameters for `transient(Circuit& W, double* x, PARAM p)`

The transient analysis is done in several steps. In a first step prior to the actual transient analysis, the differentiation index of the circuit is determined by analyzing its netlist. Then, depending on the parameter `method`, the circuit DAE is integrated either with Radau5 or Daspk3.1. During the integration, the index reduction as proposed in Chapter 4 is performed, if `ir_flag = 0`. In the following, the different steps of the transient analysis will be discussed in detail. Since the graph theoretical index determination and the index reduction are the main parts of qPsim , a greater emphasis is placed on these steps.

Graph theoretical analysis of the circuit

It is possible to perform the transient analysis in qPsim based on the circuit DAE as provided by the MNA c/f or to perform an index reduction in the case the circuit has differentiation index 2. If the parameter `ir_flag` is set to `0`, then qPsim will call the functions `int CVloopCheck(Circuit& W, std::map<std::string, Loop>)` and `int LIcutsetCheck(Circuit& W, std::map<std::string, Cutset>)` to determine the index.

The class `std::map` is an associative container which is part of the C++ Standard Template Library [69, 74]. The container is used to hold the information about all CV loops or LI cutsets which are found by `CVloopCheck` or `LIcutsetCheck`, respectively. The class `Loop` is used to store the list of capacitances and the list of voltage sources that belong to a specific CV loop. Similarly, the class `Cutset` stores the list of inductances and the list of current sources of a specific LI cutset.

The function `CVloopCheck` uses Algorithm 4 to search for 2-connected components of the subgraph $\mathfrak{G}_C$ of the network graph $\mathfrak{G}$. If a 2-connected component is found, then the component is checked for voltage sources. If it contains at least one voltage source, then the component also contains a CV loop. In this case, a set of fundamental CV loops for the component is determined by Algorithm 5. For each CV loop, the capacitance that defines the loop is stored separately in the corresponding `Loop` object. The function returns the number of fundamental CV loops that are contained in the circuit or `-1` if the function detected a loop of voltage sources.

The function `LIcutsetCheck` uses Algorithm 3 to determine a set of fundamental LI cutsets. Again, the inductance that defines an LI cutset is stored separately in the corresponding `Cutset` object. Analogously to `CVloopCheck`, the function returns the number of fundamental LI cutsets that have been found or `-1` if the function detected a cutset of current sources.

The index check needs only to be done once prior to the actual solution of the circuit DAE. The information about the index of the DAE and the CV loops and LI cutsets contained in the circuit are stored and used to reduce the index if necessary. Note that neither `CVloopCheck` nor `LIcutsetCheck` check the circuit for controlled sources that violate the conditions given in Appendix I.

Index reduction

If both `CVloopCheck` and `LIcutsetCheck` return `0`, then the circuit DAE is considered to have differentiation index 1. If either of the functions returns a value greater than 0, then the differentiation index is considered to be 2 and an index reduction is performed if `ir_flag = 0`. In this case, two new lists of `residual` objects are created that are used to model the additional functions

$$\mathbf{f}_C(t,\mathbf{x},\dot{\mathbf{x}}) := \mathbf{F}_C\frac{d}{dt}\mathbf{q} + \widehat{\mathbf{C}}(\widehat{\mathbf{A}}_C^T\mathbf{e})\mathbf{M}_{V,ind}\frac{d}{dt}\mathbf{v}_{ind}(t) \tag{5.2a}$$

and

$$\mathbf{f}_L(t,\mathbf{x},\dot{\mathbf{x}}) := \mathbf{F}_{L,1}\frac{d}{dt}\mathbf{\Phi} + \mathbf{F}_{L,2}\mathbf{A}_L^T\mathbf{e} + \widehat{\mathbf{L}}(\widehat{\mathbf{i}}_L)\mathbf{S}_{I,ind}\frac{d}{dt}\mathbf{i}_{ind}(t), \tag{5.2b}$$

with $\mathbf{x}^T = \begin{bmatrix}\mathbf{q}^T & \mathbf{\Phi}^T & \mathbf{e}^T & \mathbf{i}_V^T & \mathbf{i}_L^T\end{bmatrix}$. Both functions are evaluated by `getCVeq` and `getLIeq`. The functions `hcliFcn` and `hccvFcn` then return the actual function values $\mathbf{f}_C(t^*,\mathbf{x}^*,\dot{\mathbf{x}}^*)$ and $\mathbf{f}_L(t^*,\mathbf{x}^*,\dot{\mathbf{x}}^*)$ at t^*,$\mathbf{x}^*$ and $\dot{\mathbf{x}}^*$. To obtain the time derivatives of the source functions $\mathbf{v}_{ind}(t)$ and $\mathbf{i}_{ind}(t)$, the independent variable t is declared dependent for the automatic differentiation. Hence, these time derivatives do not have to be supplied by the user. The function `hcMass` returns the Jacobians of (5.2a) and (5.2b) with respect to either $\mathbf{x}$ or $\dot{\mathbf{x}}$. These functions are used in the routines `RAD1_MAS`, `RAD1_JAC`, `RAD1_FCN`, `D1_JAC` and `D1_FCN` which compute the actual, index reduced DAE to be integrated by Radau5 and Daspk.1.

With the information about the capacitances and inductances that define fundamental CV loops and LI cutsets, it is possible to obtain permutations $\mathbf{\Pi}_C$ and $\mathbf{\Pi}_L$ such that

$$\mathbf{\Pi}_C\mathbf{q} = \begin{bmatrix}\widehat{\mathbf{q}}\\ \widetilde{\mathbf{q}}\\ \bar{\mathbf{q}}\end{bmatrix} \text{ and } \mathbf{\Pi}_L\mathbf{\Phi} = \begin{bmatrix}\widehat{\mathbf{\Phi}}\\ \widetilde{\mathbf{\Phi}}\\ \bar{\mathbf{\Phi}}\end{bmatrix}$$

with the corresponding incidence matrices

$$\mathbf{A}_C\mathbf{\Pi}_C^T = \begin{bmatrix}\widehat{\mathbf{A}}_C & \widetilde{\mathbf{A}}_C & \bar{\mathbf{A}}_C\end{bmatrix}$$

and

$$\mathbf{A}_L\mathbf{\Pi}_L^T = \begin{bmatrix}\widehat{\mathbf{A}}_L & \widetilde{\mathbf{A}}_L & \bar{\mathbf{A}}_L\end{bmatrix}.$$

With these permutations, we are able to transform (5.2a) and (5.2b) into

$$\mathbf{f}_C(t,\mathbf{y},\dot{\mathbf{y}}) := \frac{d}{dt}\widehat{\mathbf{q}} + \widehat{\mathbf{C}}(\widehat{\mathbf{v}}_C)\mathbf{M}_C\widetilde{\mathbf{C}}^{-1}(\widetilde{\mathbf{v}}_C)\frac{d}{dt}\widetilde{\mathbf{q}} + \widehat{\mathbf{C}}(\widehat{\mathbf{v}}_C)\mathbf{M}_{V,ind}\frac{d}{dt}\mathbf{v}_{ind}(t) \tag{5.3a}$$

and

$$\mathbf{f}_L(t,\mathbf{y},\dot{\mathbf{y}}) := \frac{d}{dt}\widehat{\mathbf{\Phi}} + \widehat{\mathbf{L}}(\widehat{\mathbf{i}}_L)\mathbf{S}_L\widetilde{\mathbf{L}}^{-1}(\widetilde{\mathbf{i}}_L)\widetilde{\mathbf{A}}_L^T\mathbf{e} + \widehat{\mathbf{L}}(\widehat{\mathbf{i}}_L)\mathbf{S}_{I,ind}\frac{d}{dt}\mathbf{i}_{ind}(t), \tag{5.3b}$$

with $\mathbf{y}^T = \begin{bmatrix}(\mathbf{\Pi}_C\mathbf{q})^T & (\mathbf{\Pi}_L\mathbf{\Phi})^T & \mathbf{e}^T & \mathbf{i}_V^T & (\mathbf{\Pi}_L\mathbf{i}_L)^T\end{bmatrix}$. If we apply the permutations to the function $\mathbf{f}_{BCR,L}(t,\mathbf{x},\dot{\mathbf{x}}) := \frac{d}{dt}\mathbf{\Phi} - \mathbf{A}_L^T\mathbf{e}$ which represents equation (3.21b), then we obtain the following splitting

$$\widehat{\mathbf{f}}_{BCR,L}(t,\mathbf{y},\dot{\mathbf{y}}) = \frac{d}{dt}\widehat{\mathbf{\Phi}} - \widehat{\mathbf{A}}_L^T\mathbf{e}, \tag{5.4a}$$

$$\widetilde{\mathbf{f}}_{BCR,L}(t,\mathbf{y},\dot{\mathbf{y}}) = \frac{d}{dt}\widetilde{\mathbf{\Phi}} - \widetilde{\mathbf{A}}_L^T\mathbf{e}, \tag{5.4b}$$

$$\bar{\mathbf{f}}_{BCR,L}(t,\mathbf{y},\dot{\mathbf{y}}) = \frac{d}{dt}\bar{\mathbf{\Phi}} - \bar{\mathbf{A}}_L^T\mathbf{e}. \tag{5.4c}$$

We see that we have to subtract (5.3b) from (5.4a) in order to perform the index reduction in the case of LI cutsets.

The situation is a bit more complex in the case of CV loops. If we compare (5.3a) with (4.34)

$$\mathbf{0} = \frac{d}{dt}\widehat{\mathbf{q}}_C + \widehat{\mathbf{C}}(\widehat{\mathbf{A}}_C^T\mathbf{e})\widetilde{\mathbf{M}}_C\widetilde{\mathbf{C}}^{-1}(\widetilde{\mathbf{A}}_C^T\mathbf{e})\widetilde{\mathbf{i}}_C + \widehat{\mathbf{C}}(\widehat{\mathbf{A}}_C^T\mathbf{e})\mathbf{M}_{V,ind}\frac{d}{dt}\mathbf{v}_{ind}(t),$$

we see that (5.3a) still depends on the derivatives of the charges $\widetilde{\mathbf{q}}$ whereas (4.34) depends on the capacitive currents $\widetilde{\mathbf{i}}_C$. This is due to the fact that the `Circuit` object does not contain these currents as unknowns. Hence, in contrast to the case of LI cutsets, it is not possible to obtain the necessary equations directly from the circuit simulator.

To implement the index reduction as proposed in Section 4.3.2, we first compute the matrix

$$\widetilde{\mathbf{A}}_{C-} - \widehat{\mathbf{A}}_C\widehat{\mathbf{C}}(\widehat{\mathbf{A}}_C^T\mathbf{e})\mathbf{M}_C\widetilde{\mathbf{C}}^{-1}(\widetilde{\mathbf{A}}_C^T\mathbf{e})$$

and multiply it with the new variables $\widetilde{\mathbf{i}}_C$. The matrices $\widehat{\mathbf{A}}_C$ and $\widetilde{\mathbf{A}}_{C-}$ can be determined with the help of the information about the capacitances which are part of CV loops and about the capacitances which define fundamental CV loops. The matrix $\widetilde{\mathbf{A}}_{C+}$ can be computed at the same time as $\widetilde{\mathbf{A}}_{C-}$. The matrix

$$\widehat{\mathbf{C}}(\widehat{\mathbf{A}}_C^T\mathbf{e})\mathbf{M}_C\widetilde{\mathbf{C}}^{-1}(\widetilde{\mathbf{A}}_C^T\mathbf{e})$$

is computed as the Jacobian of $\mathbf{f}_C(t,\mathbf{y},\dot{\mathbf{y}})$ with respect to $\frac{d}{dt}\widetilde{\mathbf{q}}$. Finally, we compute the product

$$-\widehat{\mathbf{A}}_C\widehat{\mathbf{C}}(\widehat{\mathbf{A}}_C^T\mathbf{e})\mathbf{M}_{V,ind}\frac{d}{dt}\mathbf{v}_{ind}(t).$$

With these we are able to compute

$$\begin{aligned}\mathbf{0} =& \widetilde{\mathbf{A}}_{C+}^T\frac{d}{dt}\widetilde{\mathbf{q}} + \bar{\mathbf{A}}_C^T\frac{d}{dt}\bar{\mathbf{q}} + \mathbf{A}_R\mathbf{g}(\mathbf{A}_R^T\mathbf{e},t) + \mathbf{A}_L\mathbf{i}_L + \mathbf{A}_C\mathbf{i}_V \\ &+ \left(\widetilde{\mathbf{A}}_{C-} - \widehat{\mathbf{A}}_C\widehat{\mathbf{C}}(\widehat{\mathbf{A}}_C^T\mathbf{e})\widetilde{\mathbf{M}}_C\widetilde{\mathbf{C}}^{-1}(\widetilde{\mathbf{A}}_C^T\mathbf{e})\right)\widetilde{\mathbf{i}}_C + \mathbf{A}_I i_s(*,t) \\ &- \widehat{\mathbf{A}}_C\widehat{\mathbf{C}}(\widehat{\mathbf{A}}_C^T\mathbf{e})\mathbf{M}_{V,ind}\frac{d}{dt}\mathbf{v}_{ind}(t).\end{aligned}$$

Moreover, to compensate for the additional variables, the equation

$$\mathbf{0} = -\frac{d}{dt}\widetilde{\mathbf{q}} + \widetilde{\mathbf{i}}_C$$

is added to the system. The system that results in this way is equivalent to the system (4.38). However, QPSIM does not allow the evaluation of the charge conservation laws (4.38e) which depend on the node potentials $\mathbf{e}_0$ of additional nodes that were not part of the original circuit. But since a closer look at (4.38) reveals that these node potentials only occur in equations (4.38e) and

$$\mathbf{0} = \widetilde{\mathbf{A}}_{C-}^T \mathbf{e} + \mathbf{e}_0, \tag{5.5}$$

the use of the original charge conservation laws

$$\mathbf{0} = \mathbf{q} - \mathbf{q}_C(\mathbf{A}_C^T e)$$

simply means that we solve (5.5) for $\mathbf{e}_0$ and insert the result into (4.38e). This algebraic operation does not change the differentiation index. Therefore, the system that is generated in this way also has differentiation index 1.

Solving the circuit DAE

As described in the previous section, the index reduction is done inside the routines `RAD1_MAS`, `RAD1_JAC`, `RAD1_FCN`, `D1_JAC` and `D1_FCN` which are then passed to RADAU5 and DASPK3.1. Both solvers are not altered.

The parameters for RADAU5 are set such that the same tolerances are used for all variables and no variable is excluded from the error test. For stepsize control the standard controller is used (cf. [38], page 124). The nonlinear system that arises after the discretization of the DAE is solved by a simplified Newton method. As starting value for the Newton iteration the last computed value is used.

The parameters for DASPK3.1 are set such that again the same tolerances for all variables are used and no variables are excluded from error tests. In addition, the code is told that it is possible to integrate the system past the last step `params.t_end`. Note that this may not be true for all circuit DAEs.

If the user specified `params.con_inv = 0`, then QPSIM will try to compute consistent initial values from an initial guess which the user has to provide. The computation is done by the function `conival` which calls NLSCON which is a nonlinear least square solver [66]. NLSCON tries to solve the underdetermined system of nonlinear equations

$$\widetilde{\mathbf{Q}}\widetilde{\mathbf{y}} + \widetilde{\mathbf{f}}(t, \widetilde{\mathbf{x}}) = \mathbf{0},$$

for $\widetilde{\mathbf{y}}$ and $\widetilde{\mathbf{x}}$, where

$$\widetilde{\mathbf{Q}}\frac{d}{dt}\widetilde{\mathbf{x}} + \widetilde{\mathbf{f}}(t, \widetilde{\mathbf{x}}) = \mathbf{0},$$

is the index reduced system. The initial values will be computed with a tolerance set to 10^{-8}. NLSCON offers the option to place constraints on the least square solution. This could be used to allow the user to fix certain initial values. However, in the current implementation this feature is not used.

5.2. Numerical results

The examples presented here comprise two linear circuits and the NAND gate benchmark [1] with the level B MOSFET model. All of the circuits presented in this section yield MNA c/f equations which have differentiation index 2.

In the following, we will discuss the performance and efficiency of the index reduction which has been implemented in qPsim . All examples have been simulated with qPsim both with and without index reduction. As discretization methods for the DAEs both Radau5 and Daspk3.1 have been used. Since the object oriented approach used in qPsim creates an overhead for the handling of objects during runtime, the results are difficult to compare to any results obtained by other solvers.

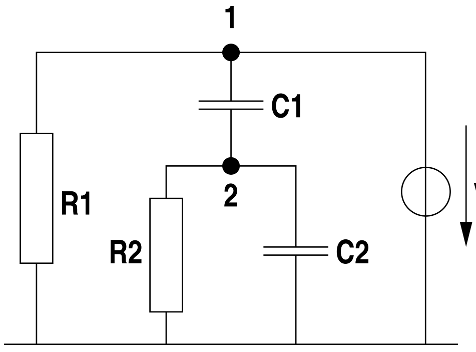


Figure 5.6.: Linear circuit with one CV loop

5.2.1. Linear circuit with one CV loop

The circuit in Figure 5.6 has already been examined in Example 4.11. For the simulation we will use $C_1 = C_2 = 1$, $R_1 = R_2 = 1$ and $v(t) = \sin(100t)$. With these parameters, the MNA c/f yields the DAE (5.6).

$$\begin{aligned}
0 &= \frac{d}{dt}q_1 + e_1 + i_V, \\
0 &= \frac{d}{dt}q_2 - \frac{d}{dt}q_1 + e_2, \\
0 &= e_1 - \sin(100t), \\
0 &= q_1 - e_1 + e_2, \\
0 &= q_2 - e_2.
\end{aligned} \tag{5.6}$$

It is possible to derive an exact analytical solution (5.7) for this example. With the consistent initial values $q_1(0) = q_2(0) = e_1(0) = e_2(0) = 0$ and $i_V(0) = -50$ the analytical solution is given by

$$
\begin{aligned}
q_1(t) &= e_1(t) - e_2(t),\\
q_2(t) &= e_2(t),\\
e_1(t) &= \sin(100t),\\
e_2(t) &= \frac{100}{40001}\cos(100t) + \frac{20000}{40001}\sin(100t) - \frac{100}{40001}e^{-\frac{1}{2}t},\\
i_V(t) &= -\frac{2000100}{40001}\cos(100t) - \frac{50001}{40001}\sin(100t) + \frac{50}{40001}e^{-\frac{1}{2}t}.
\end{aligned}
\tag{5.7}
$$

This solution has been used to compute the absolute errors in each variable. A numerical solution has been computed with both RADAU5 and DASPK3.1 on the interval $[0, 0.1]$ with relative and absolute accuracy ranging from 10^{-3} to 10^{-12}. For the integration both the original formulation of the circuit DAE and the index reduced formulation have been used. If no index reduction was performed, DASPK3.1 failed to converge for accuracies smaller than 10^{-5}. The achieved accuracies with respect to each unknown have been computed using the Euclidean norm of the vector absolute errors for the respective unknown. Figure 5.7, 5.8 and 5.9 show the results for q_1, e_1 and i_V. Note that in this example the current i_V is the variable which causes the higher index of the DAE (5.6). The figures suggest that both solvers perform better if the index reduction method proposed in Section 4.3 is applied. The effect is most noticeable in the efficiency with respect to i_V (cf. Figure 5.9).

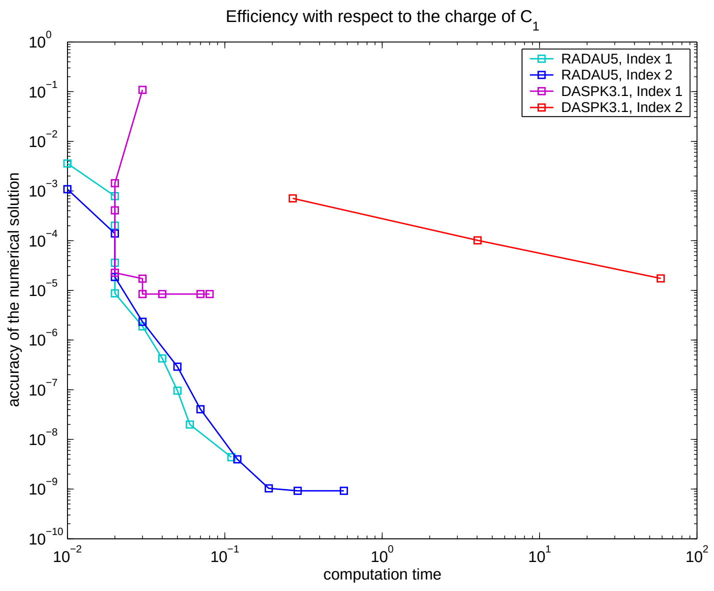


Figure 5.7.: Efficiency with respect to the charge of C_1

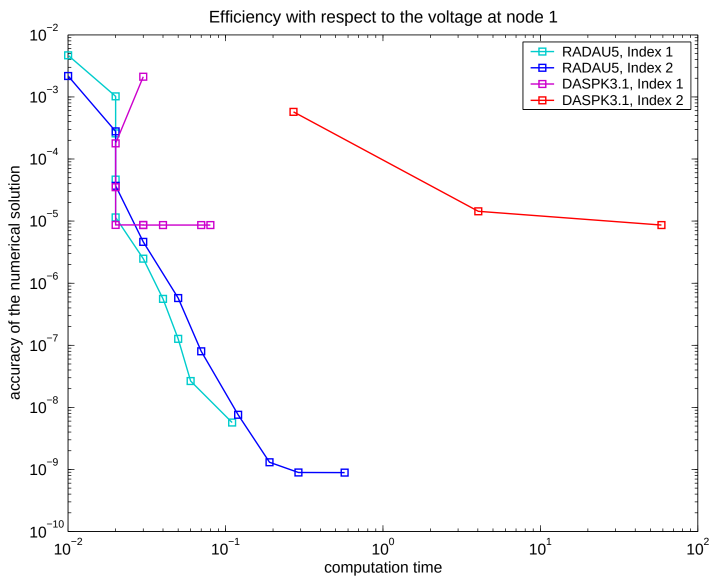


Figure 5.8.: Efficiency with respect to the voltage at node 1

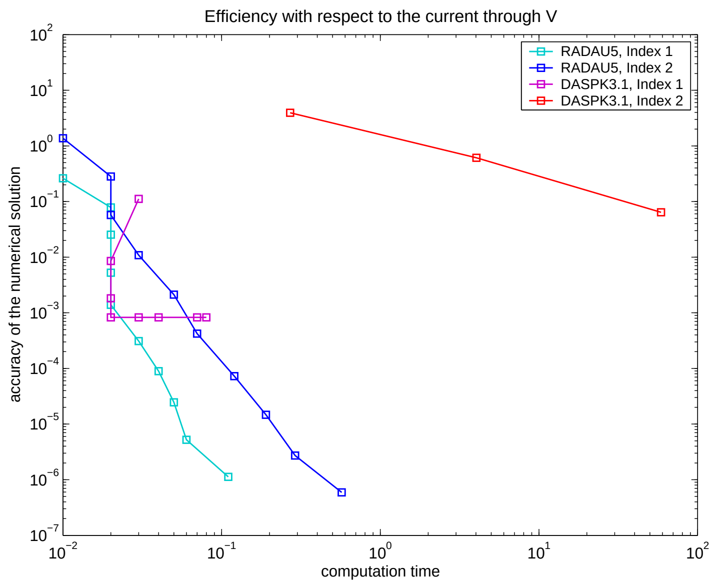


Figure 5.9.: Efficiency with respect to the current through the voltage source

5.2.2. Linear circuit with one LI cutset

In this example, we consider the circuit in Figure 5.10. For $R_1 = R_2 = 1$, $L_1 = L_2 = 1$ and $i(t) = \sin(100t)$ we obtain (5.8) from MNA c/f. Due to the LI cutset that consists of the inductances L_1 and L_2 and the current source, the circuit DAE (5.8) has differentiation index 2.

$$\begin{aligned}
0 &= 2e_1 - e_2 + i_{L1},\\
0 &= -e_1 + e_2 + i_{L2},\\
0 &= -i_{L1} - i_{L2} + i(t),\\
0 &= \frac{d}{dt}\phi_1 - e_1 + e_3,\\
0 &= \frac{d}{dt}\phi_2 - e_2 + e_3,\\
0 &= \phi_1 - i_{L1},\\
0 &= \phi_2 - i_{L2}.
\end{aligned} \tag{5.8}$$

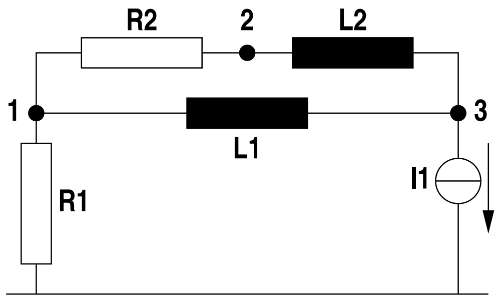


Figure 5.10.: Linear circuit with one LI cutset

As for the previous example, it is possible to compute an analytical solution for given initial values. Here the initial values were chosen to be $\phi_1(0) = \phi_2(0) = e_1(0) = e_2(0) = i_{L1}(0) = i_{L2}(0) = 0$ and $e_3(0) = -50$. With these values, the analytical solution of (5.8) is given by

$$\begin{aligned}
\phi_1(t) &= -\frac{100}{40001}\cos(100t) + \frac{20001}{40001}\sin(100t) + \frac{100}{40001}e^{-\frac{1}{2}t},\\
\phi_2(t) &= \frac{100}{40001}\cos(100t) + \frac{20000}{40001}\sin(100t) - \frac{100}{40001}e^{-\frac{1}{2}t},\\
e_1(t) &= -\sin(100t),\\
e_2(t) &= e_1(t) - \phi_2(t),\\
e_3(t) &= \frac{e_1(t) + e_2(t)}{2} - 50\cos(100t),\\
i_{L1}(t) &= \phi_1(t),\\
i_{L2}(t) &= \phi_2(t).
\end{aligned} \tag{5.9}$$

Numerical solutions have been computed by both Radau5 and Daspk3.1 with and without index reduction for absolute and relative accuracies ranging from 10^{-3} to 10^{-12}. For accuracies smaller than 10^{-5} Daspk3.1 failed to converge if no index reduction was performed. The exact solution (5.9) was used to compute the absolute errors in the unknowns. As a measure for the accuracies achieved by the codes the Euclidean norm of the vectors of absolute errors in each variable has been computed. Figures 5.11, 5.12 and 5.13 display the results for the voltages at node 1 and 3 and the current through L_1. Note that in this example the voltage e_3 at node 3 is the variable that causes the higher differentiation index of DAE (5.8). Again, the performance of both solvers increased if the index reduction method from Section 4.3 was applied to the DAE (5.8). As in the previous example, this effect is most noticeable in the unknown that causes the higher differentiation index, i.e. the voltage e_3 (cf. Figure 5.12).

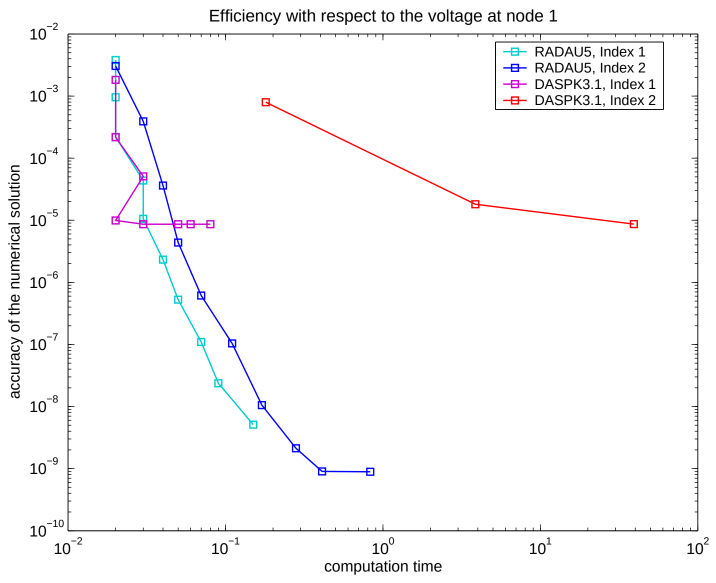


Figure 5.11.: Efficiency with respect to the voltage at node 1

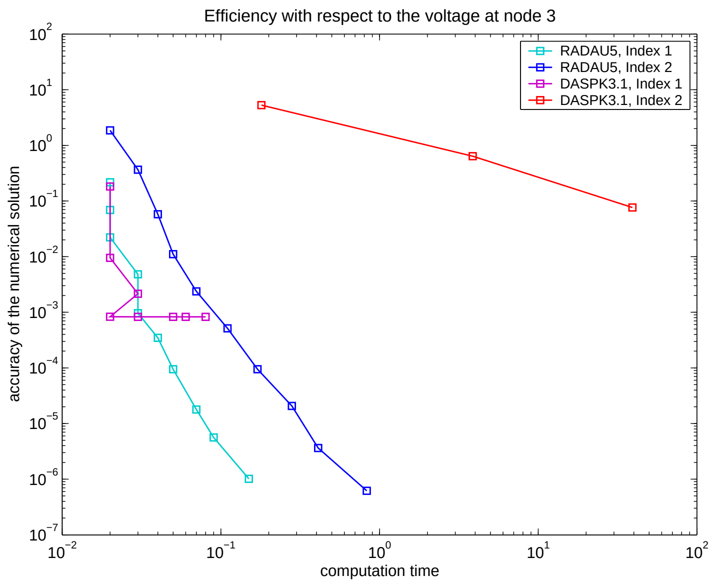


Figure 5.12.: Efficiency with respect to the voltage at node 3

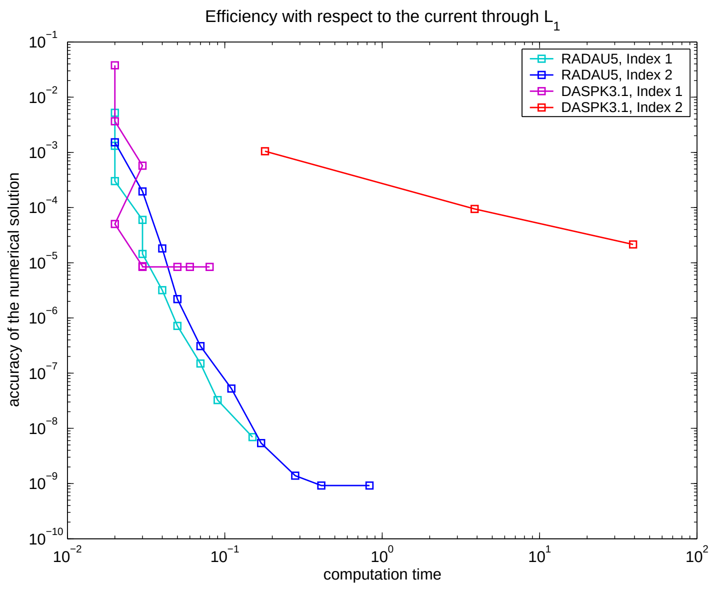


Figure 5.13.: Efficiency with respect to the current through L_1

5.2.3. NAND gate with level B MOSEFT model

In Example 4.12, we introduced the NAND gate benchmark [1]. Here, the NAND gate is simulated with QPSIM. For the simulation the MOSFETs in the circuit have been modeled with the level B MOSFET model.

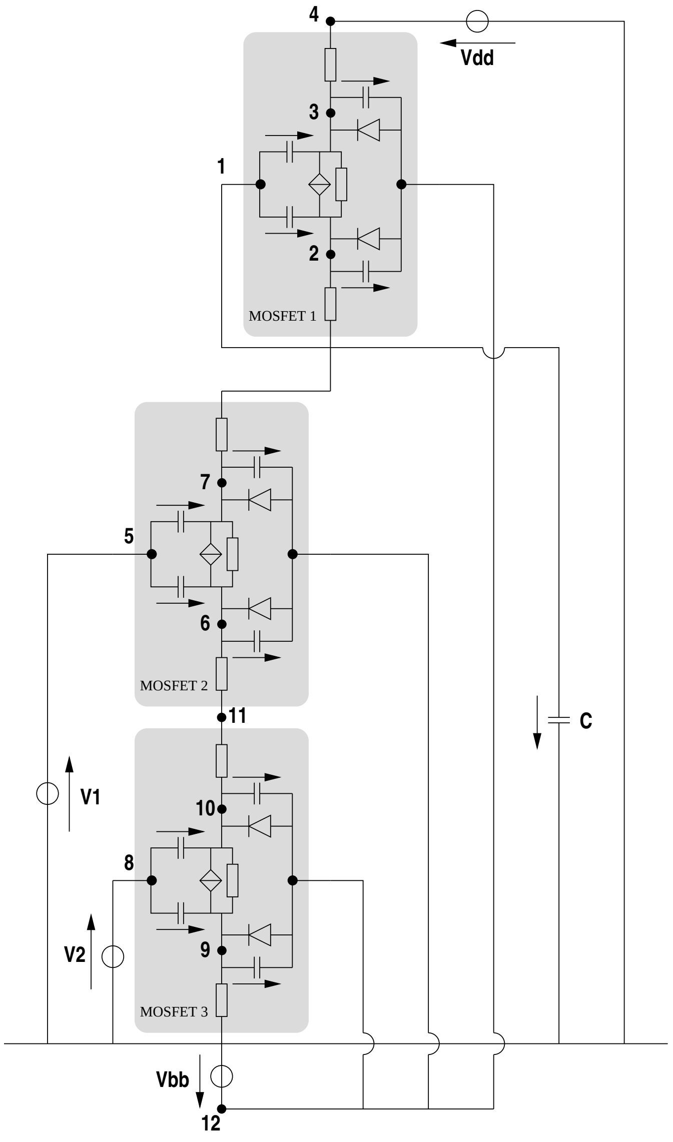


Figure 5.14.: NAND gate

While the level A MOSFET model uses linear bulk capacitances, the level B model uses nonlinear bulk capacitances for which the charges are given by

$$q_{db}(v) = q_{sb}(v) = \begin{cases} C_0 \cdot \Phi_B \cdot \left(1 - \sqrt{1 - \frac{v}{\Phi_B}}\right) & \text{for } v \leq 0, \\ C_0 \cdot \left(1 + \frac{v}{4\Phi_B}\right) \cdot v & \text{for } v > 0, \end{cases}$$

with $C_0 = 0.24 \cdot 10^{-13}F$ and $\Phi_B = 0.87V$. The gate capacitances both have a capacitance value of $C_{gs} = C_{gd} = 0.6 \cdot 10^{-13}F$. The contact resistances R_s and R_d are given by $R_s = R_d = 4\Omega$. To model the fact that the gate is isolated from the drain-source channel, the resistance R_{sd} is given by $R_{sd} = 10^{15}\Omega$.

The voltage controlled current source models the technology dependent current gain of the MOSFET. For the level B MOSFET model this current gain is given by

$$i_{ds}(v_{ds}, v_{gs}, v_{bs}, v_{gd}, v_{bd}) = \begin{cases} 0 & \text{for } U \leq 0, \\ -\beta\left(1 + \delta v_{ds}\right) U^2 & \text{for } 0 < U \leq v_{ds}, \\ -\beta v_{ds}\left(1 + \delta v_{ds}\right)\left(2U - v_{ds}\right) & \text{for } v_{ds} < U \end{cases}$$

with $U = v_{gs} - \left(U_{T0} + \gamma \cdot \left(\sqrt{\Phi - v_{bs}} - \sqrt{\Phi}\right)\right)$ if $v_{ds} > 0$ and $\Phi > v_{bs}$,

$$i_{ds}(v_{ds}, v_{gs}, v_{bs}, v_{gd}, v_{bd}) = 0$$

if $v_{ds} = 0$ and

$$i_{ds}(v_{ds}, v_{gs}, v_{bs}, v_{gd}, v_{bd}) = \begin{cases} 0 & \text{for } U \leq 0, \\ -\beta\left(1 - \delta v_{ds}\right) U^2 & \text{for } 0 < U \leq -v_{ds}, \\ -\beta v_{ds}\left(1 - \delta v_{ds}\right)\left(2U - v_{ds}\right) & \text{for } -v_{ds} < U \end{cases}$$

with $U = v_{gd} - \left(U_{T0} + \gamma \cdot \left(\sqrt{\Phi - v_{bd}} - \sqrt{\Phi}\right)\right)$ if $v_{ds} < 0$ and $\Phi > v_{bd}$. The values for the parameters U_{T0}, β, γ, δ and Φ determine the behavior of the MOSFET. They can be found in Appendix II. The load capacitance C is given by $C = 0.5 \cdot 10^{-13}F$.

The NAND gate consists of three MOSFETs. MOSFET1 is a depleting MOSFET. MOSFET2 and MOSFET3 are enhancing MOSFETs. Roughly speaking, enhancing means that the drain-source channel opens if the gate voltage rises above a technology dependent threshold. For a depleting MOSFET, the drain-source channel is open until the gate voltage reaches a technology dependent threshold.

The drain voltage of MOSFET1 is constant at $V_{DD} = 5V$, the bulk voltages of all MOSFETs are constant at $V_{BB} = -2.5V$. The input signals are given by the voltage sources V_1 and V_2. If both signals are at $5V$ (i.e. HIGH), then the response at node 1 is at $0V$ (i.e. LOW). Otherwise the response is HIGH. The input sources have been modeled both with piecewise linear functions and with functions in $\mathcal{C}^1(\mathbb{I})$ where $\mathbb{I} = [0, 8 \cdot 10^{-8}]$ is the integration interval. Figure 5.15 shows the numerical solution for the NAND gate with piecewise linear input signals. The solution has been computed with RADAU5 with index reduction and a prescribed accuracy of 10^{-15}.

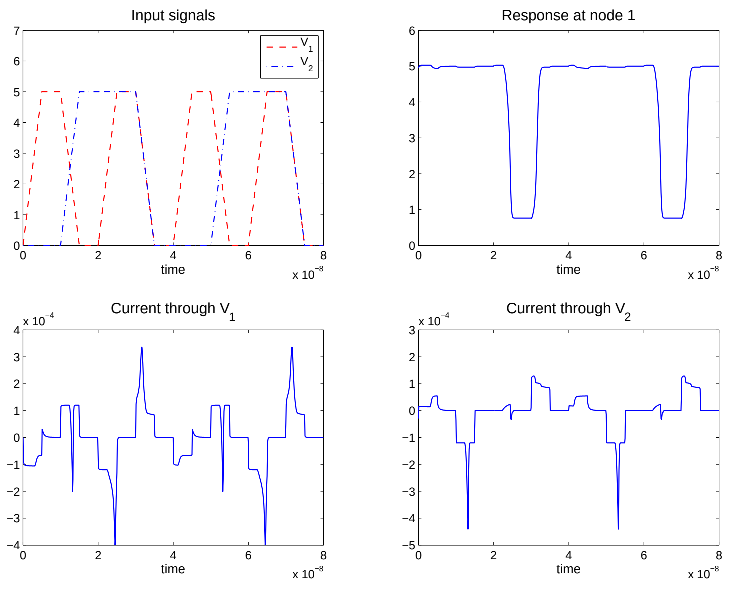


Figure 5.15.: Reference solution for the NAND gate example with piecewise linear input signals

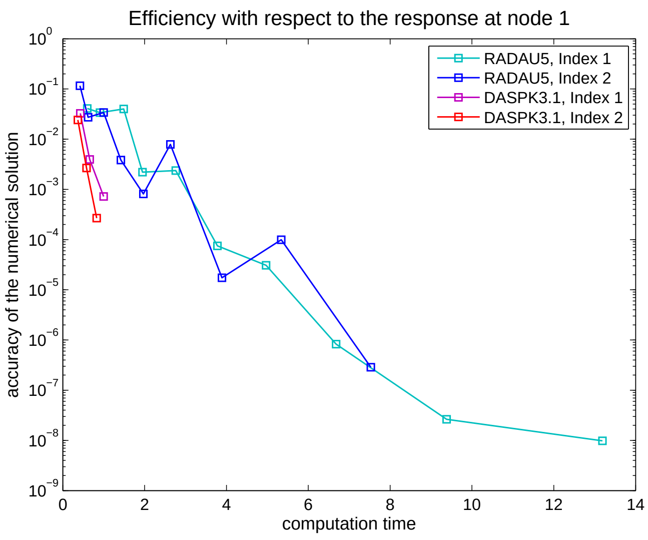


Figure 5.16.: Efficiency with respect to e_1 for the NAND gate example with piecewise linear input signals

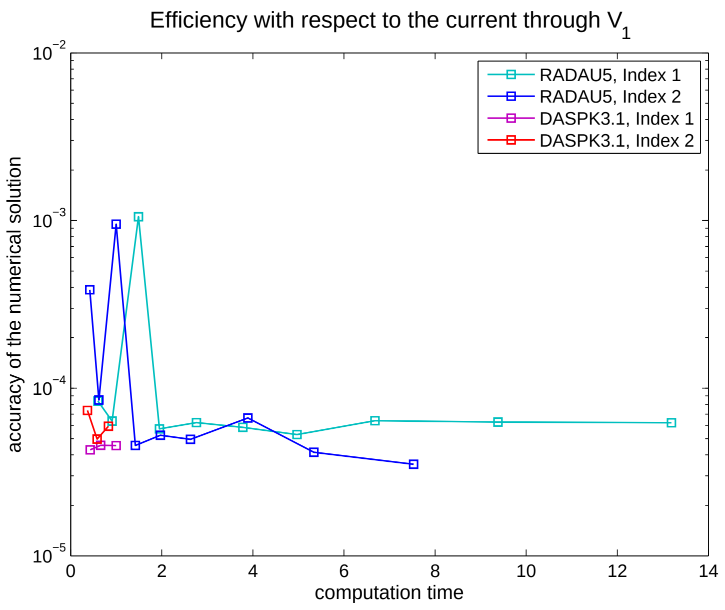


Figure 5.17.: Efficiency with respect to i_1 for the NAND gate example with piecewise linear input signals

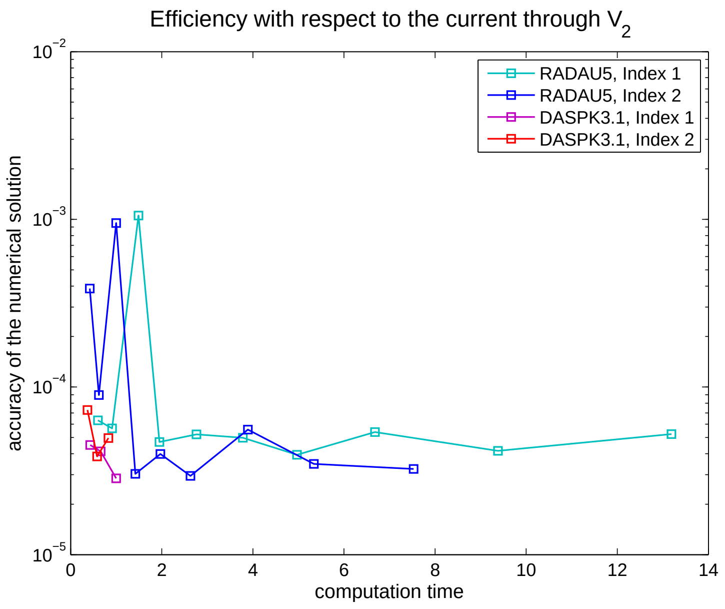


Figure 5.18.: Efficiency with respect to i_2 for the NAND gate example with piecewise linear input signals

To measure the efficiency of the index reduction method implemented in QPSIM, a numerical solution for the circuit DAE of the NAND gate has been computed with RADAU5 and DASPK3.1 for accuracies between 10^{-3} and 10^{-12}. The computation was done with and without index reduction. The initial values have been taken from [26] and corrected by `conival` (cf. Table 5.2). The results are compared to a reference solution which has been computed by RADAU5 with index reduction. The tolerance for the reference solution has been set to 10^{-15}.

$q = 2.48702e^{-13}$		
$q_{gd_1} = -1.53975e^{-15}$	$q_{gd_2} = -2.98424e^{-13}$	$q_{gd_3} = 2.3029e^{-15}$
$q_{gs_1} = -1.81303e^{-17}$	$q_{gs_2} = 2.32257e^{-15}$	$q_{gs_3} = -2.39079e^{-27}$
$q_{sb_1} = 5.67893e^{-13}$	$q_{sb_2} = 5.64588e^{-13}$	$q_{sb_3} = 1.00869e^{-13}$
$q_{db_1} = 5.64665e^{-13}$	$q_{db_2} = 1.0085e^{-13}$	$q_{db_3} = 1.03103e^{-13}$
$e_1 = 4.97404$	$e_2 = 4.97434$	$e_3 = 4.9997$
$e_4 = 5.0$	$e_5 = -3.12224e^{-39}$	$e_6 = -0.0387095$
$e_7 = 4.97373$	$e_8 = 7.84801e^{-44}$	$e_9 = 3.98465e^{-14}$
$e_{10} = -0.0383817$	$e_{11} = -0.0385456$	$e_{12} = -2.5$
$i_{DD} = -7.5543e^{-05}$	$i_{BB} = 9.98898e{-}05$	
$i_1 = -3.58793e^{-25}$	$i_2 = -2.43469e^{-05}$	

Table 5.2.: Initial values for the NAND gate with level B MOSFET and piecewise linear input signals

For piecewise linear input functions, DASPK3.1 failed to integrate the circuit DAE for accuracies smaller than 10^{-5}, whether the index reduction was performed or not. In all cases, the nonlinear solver of DASPK3.1 failed to converge. This suggests that DASPK3.1 may have some difficulties with the discontinuities in the derivatives of the input signals. RADAU5 failed to converge for prescribed accuracies of 10^{-10} and 10^{-12}. If the index reduction was performed, then RADAU5 successfully integrated the circuit DAE for all prescribed accuracies. Figure 5.16, 5.17 and 5.18 show the efficiencies with respect to the response at node 1 and the currents through the input sources in the case of piecewise linear input functions. We see, that the solution computed with DASPK3.1 reaches higher accuracies if the index reduction is performed, whereas the accuracies for the solutions computed without index reduction are independent of the prescribed accuracies. The same is true for the numerical solutions computed with RADAU5 for prescribed accuracies smaller than 10^{-7}. The bad performance of RADAU5 with index reduction for lower accuracies is again due to the discontinuous derivatives of the input signals.

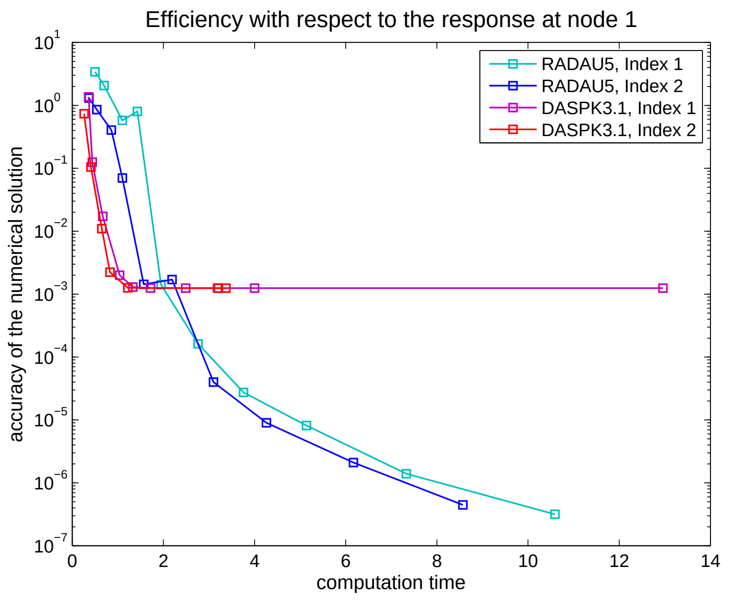


Figure 5.19.: Efficiency with respect to e_1 for the NAND gate example with smooth input signals

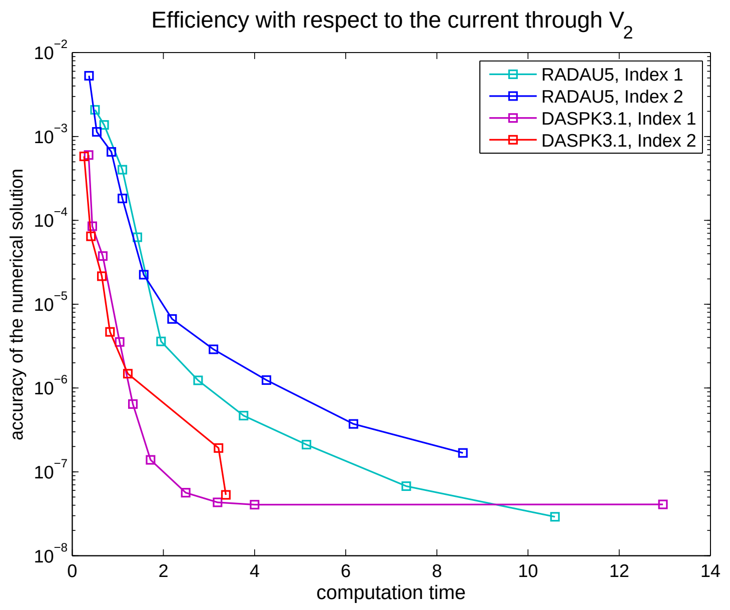


Figure 5.21.: Efficiency with respect to i_2 for the NAND gate example with smooth input signals

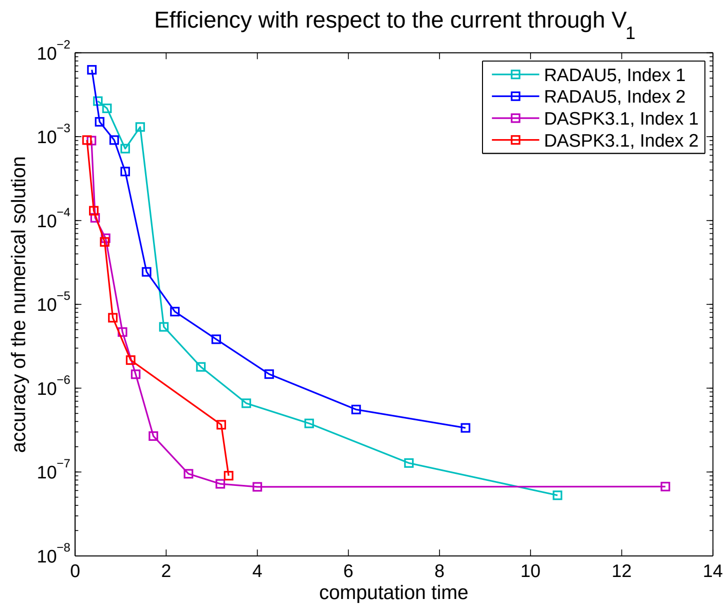


Figure 5.20.: Efficiency with respect to i_1 for the NAND gate example with smooth input signals

To exclude the effects caused by the discontinuities, the parts of the input functions that model the switching of a signal from LOW to HIGH and vice versa have been replaced by the cubic polynomials

$$p_1(x) = -\frac{2}{25}x^3 + \frac{3}{5}x^2, \qquad \text{if the signal switches from LOW to HIGH,}$$
$$p_2(x) = \frac{2}{25}x^3 - \frac{3}{5}x^2 + 5, \qquad \text{if the signal switches from HIGH to LOW,}$$

for $0 \leq x \leq 5$. The switching times are the same as for the original piecewise linear input functions. The input signals obtained in this way have continuous derivatives. The initial values from [26] were taken as an initial guess to determine the set of consistent initial values shown in Table 5.3. With the new input functions, DASPK3.1 was able to compute a numerical solution without index reduction for prescribed accuracies ranging from 10^{-3} to 10^{-9}. With index reduction, it was possible to obtain a numerical solution with DASPK3.1 for all prescribed accuracies. RADAU5 successfully computed a solution both with and without index reduction for all prescribed accuracies. The numerical solutions were compared to the solution computed by RADAU5 with index reduction at a prescribed accuracy of 10^{-15}. Figures 5.19, 5.20 and 5.21 show the efficiencies with respect to the response at node 1 and the currents through the input sources V_1 and V_2. It can be seen that both solvers perform much better for all prescribed accuracies if the code is told to apply the index reduction method.

$q = 2.5e^{-13}$		
$q_{gd_1} = 3.01207e^{-25}$	$q_{gd_2} = -3.0e^{-13}$	$q_{gd_3} = 2.32257e^{-15}$
$q_{gs_1} = 1.19071e^{-27}$	$q_{gs_2} = 2.32257e^{-15}$	$q_{gs_3} = -2.39071e^{-27}$
$q_{sb_1} = 5.67931e^{-13}$	$q_{sb_2} = 5.67931e^{-13}$	$q_{sb_3} = 1.0085e^{-13}$
$q_{db_1} = 5.67931e^{-13}$	$q_{db_2} = 1.0085e^{-13}$	$q_{db_3} = 1.03103e^{-13}$
$e_1 = 5.0$	$e_2 = 5.0$	$e_3 = 5.0$
$e_4 = 5.0$	$e_5 = -1.93398e^{-50}$	$e_6 = -0.0387095$
$e_7 = 5.0$	$e_8 = -3.9819e^{-45}$	$e_9 = 3.98452e^{-14}$
$e_{10} = -0.0387095$	$e_{11} = -0.0387095$	$e_{12} = -2.5$
$i_{DD} = 2.49613e^{-14}$	$i_{BB} = -4.98238e^{-14}$	
$i_1 = 1.50275e^{-16}$	$i_2 = 4.72672e^{-23}$	

Table 5.3.: Initial values for the NAND gate with level B MOSFET and smooth input signals

6. Summary

The main goal of this thesis has been to develop an index reduction method that is suitable for DAEs that arise from the modeling of a circuit by means of the classical or the charge-oriented Modified Nodal Analysis. To this end, an introduction to the theory of general nonlinear and quasi-linear DAEs was given in Chapter 2. Also in Chapter 2, the index reduction method by minimal extension was presented which was the basis of the development of structured index reduction methods for circuit DAEs.

In order to be able to generate the DAE which describes the behaviour of a given circuit in a systematic way, concepts from graph theory have used. Moreover, it is possible to determine the differentiation index of a circuit DAE by examining the corresponding circuit graph. If this graph contains CV loops or LI cutsets, then the DAE resulting from either variant of the Modified Nodal Analysis has differentiation index 2. These results along with the definitions of the graph theoretical concepts involved have been summarized in Chapter 3.

Chapter 4 presented the main results. We have shown that under the assumption that both the capacitance matrix $\mathbf{C}(\mathbf{v}_C)$ and the inductance matrix $\mathbf{L}(\mathbf{i}_L)$ are diagonal matrices, it is possible to reduce the differentiation index of a circuit DAE in such a way that the index reduced system still has the structural properties of the original circuit DAE. For the index reduction method for DAEs from charge-oriented Modified Nodal Analysis, it was possible to prove that the index reduced system still has a properly stated leading term. It also has been shown that both index reduction methods from Chapter 4 can be interpreted as modifications to the original circuit which allows for an easy application of the proposed methods to existing circuit simulation software.

Finally, Chapter 5 presented an academic circuit simulator which includes the index reduction method proposed for circuit DAEs from the charge-oriented Modified Nodal Analysis. The examples which have been considered show that the numerical solvers Daspk3.1 and Radau5 which have been used to discretize the circuit DAE, perform better if the index reduction method is applied.

One major drawback of the proposed index reduction methods is that they focus on circuits for which the capacitance and the inductance matrix are diagonal matrices. For most circuits arising in industrial applications this assumption will not be fulfilled. To apply the proposed methods successfully to industrial circuits, a generalization of the results of this thesis is under investigation. However, it is not clear as of yet if there exists an interpretation of the generalized method in terms of modifications to the circuit.

I. Controlled sources

The following conditions for controlled sources in lumped circuits can be found in [25], [27] or [28].

I.1. Controlled voltage sources

The controlled voltage sources are not allowed to be part of CV loops. Their controlling elements have to fulfill the conditions given in Table I.1 for voltage controlled voltage sources (VCVS) and Table I.2 for current controlled voltage sources (CCVS).

The controlling voltage of a VCVS can be the voltages across

1. capacitances,
2. independent voltage sources,
3. CCVSs that are controlled by the currents through
    a) inductances,
    b) independent current sources,
    c) resistances or VCCSs for which the nodes that are incident with the controlling branch are connected by
        i. capacitances,
        ii. independent voltage sources,
        iii. paths that contain only elements that are described in (3(c)i) and (3(c)ii),
    d) branches that form a cutset with elements described in (3a), (3b) and (3c),
4. branches that form a loop with elements described in (1), (2) and (3).

Table I.1.: Conditions for voltage controlled voltage sources

The controlling currents of a CCVS can be currents through

1. inductances,
2. independent current sources,
3. resistances or VCCSs for which the controlling nodes are connected by
    a) capacitances,
    b) independent voltage sources,
    c) VCVSs for which the nodes that are incident with the controlling branch are connected by
        i. capacitances,
        ii. independent voltage sources,
        iii. paths that contain only elements that are described in (3(c)i) and (3(c)ii),
    d) paths that contain only elements described (3a), (3b) and (3c),
4. branches that form a cutset with elements described in (1), (2) and (3).

Table I.2.: Conditions for current controlled voltage sources

I.2. Controlled current sources

Each controlled current source has to fulfill at least one of the following conditions:

1. The controlled current source is not part of any LI cutset and the controlling elements fulfill the conditions given in Table I.4 for voltage controlled current sources (VCCS) and Table I.5 for current controlled current sources (CCCS).
2. There exists a path of capacitive branches that connects the nodes that are incident with the controlled current source. The controlling elements fulfill the conditions given in Table I.6 for CCCSs. The VCCSs are controlled by arbitrary voltages.
3. There exists a path of capacitances and voltage sources that connects the nodes that are incident with the controlled current source. The controlling elements fulfill the conditions given in Table I.7. The VCCSs are controlled by arbitrary voltages.

Table I.3.: Conditions for the controlled current sources

The controlling voltages of a VCCS can be voltages across

1. capacitances,
2. voltage sources,
3. branches that form a loop with branches that are described in (1) and (2).

Table I.4.: Conditions for voltage controlled current sources

The controlling current of a CCCS can be the current through

1. inductances,
2. independent current sources,
3. resistances or VCCSs for with the nodes that are incident with the CCCS are connected by
   a) capacitances,
   b) voltage sources,
   c) paths that contain only elements described in (3a) and (3b),
4. branches that form a cutset with elements described in (1), (2) and (3).

Table I.5.: Conditions for current controlled current sources in Table I.3 (1).

The controlling current of a CCCS can be the current through

1. inductances,
2. independent current sources,
3. resistances,
4. voltage sources that do not belong to any CV loops,
5. VCCSs,
6. a branch that forms a cutset with elements described in (1) – (5).

Table I.6.: Conditions for current controlled current sources in Table I.3 (2).

The controlling current of a CCCS can be the current through

1. inductances,
2. resistances,
3. independent current sources,
4. VCCSs,
5. a branch that forms a cutset with elements described in (1) – (4).

Table I.7.: Conditions for current controlled current sources in Table I.3 (3).

I.3. Proof of Theorem 4.1

Since this change only affects sources that are controlled by either capacitances or independent voltage sources in CV loops, we only need to consider this kind of sources. From Tables I.1–I.6, we see that such a controlled source is required to be controlled by either a capacitance or an independent voltage source or a path that only consists of capacitances and independent voltage sources. However, every path in $\mathfrak{G}$ that consists only of capacitances and independent voltage sources is transformed into a path of the same kind in $\mathfrak{G}'$. As the remaining controlled sources are not affected by the change, this concludes the proof. $\square$

I.4. Proof of Theorem 4.2

First, we consider the case that a source is only controlled by a capacitance in $\mathfrak{G}$ that is replaced in $\mathfrak{G}'$ by a controlled current source. Table I.8 and Table I.9 list all possible control relations for this case. Moreover, for each relation the conditions in Table I.1–I.7 that cover the relation are listed. Here, $a \leftarrow b$ means that element a is controlled by element b.

Possible control relations	conditions
VCVS ← repl. capacitance	Table I.1 (4)
VCVS ← CCVS ← VCCS ← repl. cap.	Table I.1 (3(c)iii)
CCVS ← VCCS ← VCVS ← repl. cap.	Table I.2 (3(c)iii)
CCVS ← VCCS ← repl. capacitance	Table I.2 (3d)
VCCS ← repl. capacitance	Table I.4 (3)
CCCS ← VCCS ← repl. capacitance	Table I.5 (3c)

Table I.8.: Sources controlled by replaced capacitances

The second case deals with sources that are controlled by paths of capacitances and voltage sources in $\mathfrak{G}$ which include a capacitance that is replaced in $\mathfrak{G}'$ by a controlled current source. Since this capacitance is part of a CV loop, we are able to replace its branch by a path of independent voltage sources and capacitances. Hence, the conditions in Appendix I are fulfilled, too. $\square$

I.5. Proof of Theorem 4.3

Any voltage source that replaces an inductance in a LI cutset naturally is part of a cutset that, apart from the voltage source, contains only independent current sources and inductances. Hence, those voltage sources satisfy the conditions for controlling elements. Table I.9 lists all possible cases as well as the conditions in Tables I.1-I.7 that cover them.

Possible control relations	conditions
VCVS ← CCVS ← repl. inductance	Table I.1 (3d)
CCVS ← repl. inductance	Table I.2 (4)
CCCS ← repl. inductance	Table I.5 (4),
	Table I.6 (6),
	Table I.7 (5)

Table I.9.: Source controlled by replaced inductances

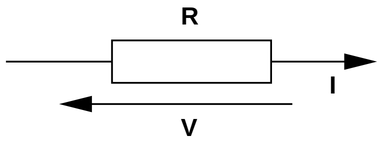
$\square$

II. Circuit elements in qPsim

II.1. Two-terminal elements

Resistance

- Graphical symbol:



- Branch constitutive relation:

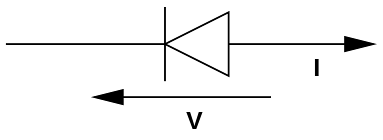
$$I = \frac{1}{R}V,\ V = RI$$

- qPsim constructor:

```
Res::Res(SubCircuit* parent, char* name, BaseNode& p, BaseNode& n,
        ad_double r);
```

Silicon diode

- Graphical symbol:



- Branch constitutive relation:

$$I = I_s\left(e^{(V/V_T)} - 1\right)$$

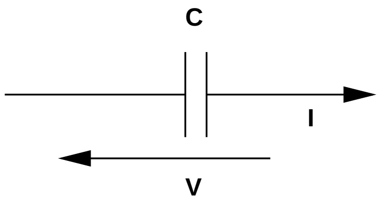
I_s	saturation current $I_s = 10^{-14}A$
$V_T = \frac{kT}{e}$	thermal voltage, $V_T \approx 25mV$
T	absolute temperature of the p-n junction
k	Boltzmann's constant
e	elementary charge

- qPsim constructor:

```
Diode::Diode(SubCircuit* parent, char* name, BaseNode& p, BaseNode& n);
```

Capacitance

- Graphical symbol:



- Branch constitutive relation:

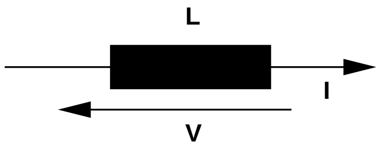
$$I = C\frac{d}{dt}V$$

- qPsim constructor:

```
Cap::Cap(SubCircuit* parent, char* name, BaseNode& p, BaseNode& n,
         ad_double c);
```

Inductance

- Graphical symbol:



- Branch constitutive relation:

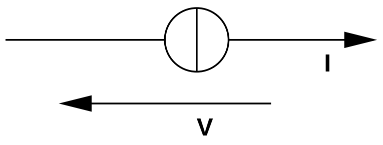
$$V = L\frac{d}{dt}I$$

- qPsim constructor:

```
Ind::Ind(SubCircuit* parent, char* name, BaseNode& p, BaseNode& n,
         ad_double l);
```

Independent current source

- Graphical symbol:



- Branch constitutive relation:

$$I = I_0,\ V\text{arbitrary}$$

- qPsim constructor:

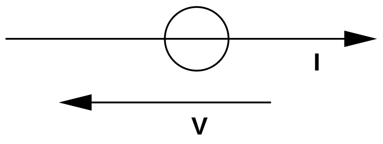
```
Isrc::Isrc(SubCircuit* parent, char* name, BaseNode& p, BaseNode& n,
           ad_double i0);
```

If the current source is time depending, i.e. $I_0 = i(t)$, then the user has to provide the method *time_stimulus(ad_double t)* for the circuit under consideration in the following way

```
void time_stimulus(ad_double t)
 {
  i.set_i( <function in t> );
 }
```

Independent voltage source

- Graphical symbol:



- Branch constitutive relation:

$$V = V_0,\ I\text{arbitrary}$$

- qPsim constructor:

```
Vsrc::Vsrc(SubCircuit* parent, char* name, BaseNode& p, BaseNode& n,
           ad_double v0);
```

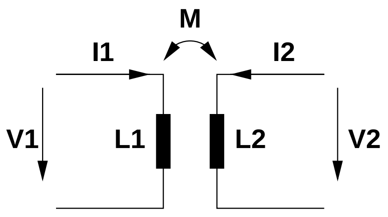
If the voltage source is time depending, i.e. $V_0 = v(t)$, then the user has to provide the method *time_stimulus(ad_double t)* for the circuit under consideration in the following way

```
void time_stimulus(ad_double t)
 {
  v.set_v( <function in t> );
 }
```

II.2. Twoport elements

Coupled indcutances – transformer

- Graphical symbol:



- Branch constitutive relation:

$$\begin{bmatrix} L_1 & M \\ M & L_2 \end{bmatrix} \frac{d}{dt} \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} - \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = 0$$

The *coefficient of coupling* K is given by

$$K = \frac{|M|}{\sqrt{L_1 L_2}}, \ 0 \leq K < 1.$$

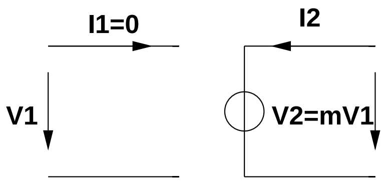
If $K = 1$, then the transformer is *ideal.*

- qPsim constructor:

```
Cpl::Cpl(SubCircuit* parent, char* name, Ind& L1, Ind& L2,
         ad_double k);
```

Voltage controlled voltage source

- Graphical symbol:



- Branch constitutive relation:

$$I_1 = 0,\ V_2 = mV_1$$

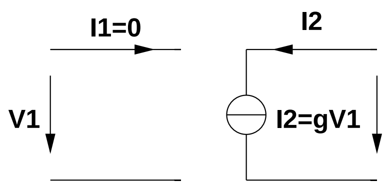
The constant m is called *voltage gain*.

- qPsim constructor:

```
VCVS::VCVS(SubCircuit* parent, char* name,
           BaseNode& p1, BaseNode& n1,
           BaseNode& p2, BaseNode& n2,
           ad_double m);
```

Voltage controlled current source

- Graphical symbol:



- Branch constitutive relation:

$$I_1 = 0,\ I_2 = gV_1$$

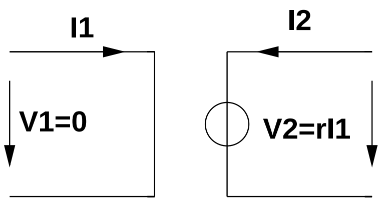
The constant g is called *transconductance*.

- qPsim constructor:

```
VCIS::VCIS(SubCircuit* parent, char* name,
           BaseNode& p1, BaseNode& n1,
           BaseNode& p2, BaseNode& n2,
           ad_double g);
```

Current controlled voltage source

- Graphical symbol:



- Branch constitutive relation:

$$V_1 = 0, \ V_2 = rI_1$$

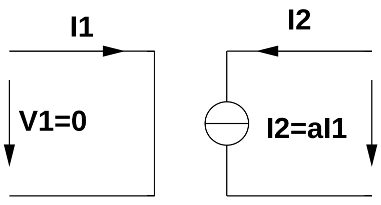
  The constant r is called *transresistance.*

- QPSIM constructor:

```
ICVS::ICVS(SubCircuit* parent, char* name,
           BaseNode& p1, BaseNode& n1,
           BaseNode& p2, BaseNode& n2,
           ad_double r);
```

Current controlled current source

- Graphical symbol:



- Branch constitutive relation:

$$V_1 = 0, \ I_2 = aI_1$$

  The constant a is called *current gain.*

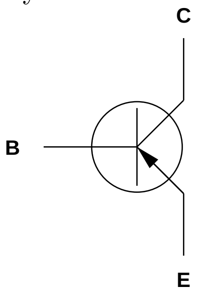
- QPSIM constructor:

```
ICIS::ICIS(SubCircuit* parent, char* name,
           BaseNode& p1, BaseNode& n1,
           BaseNode& p2, BaseNode& n2,
           ad_double a);
```

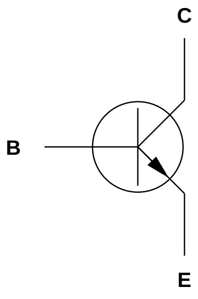
II.3. Multiport elements

Bipolar transistor

- Graphical symbol:



pnp-transistor



npn-transistor

- QPSIM constructor:

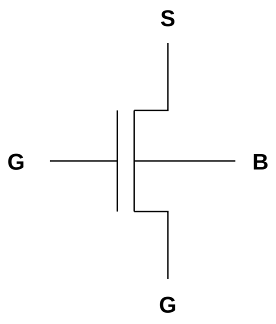
```
Bjt::Bjt(SubCircuit* parent, char* nm,
         BaseNode& c, BaseNode& b, BaseNode& e,
         double type);
```

- Parameter:

type=1	npn-transistor
type=-1	pnp-transistor

MOSFET transistor - Level 1

- Graphical symbol:



- QPSIM constructor:

```
Mosfet::MosfetL1(SubCircuit* parent, char* name,
                 BaseNode& drain, BaseNode& gate,
                 BaseNode& source, BaseNode& bulk,
                            double ut0, double beta,
                            double gamma, double phi,
                            double delta);
```

- Parameter:

	enhancement type	depletion type
U_{T_0}	$0.8V$	$-2.43V$
β	$1.748 \cdot 10^{-3} A/V^2$	$5.35 \cdot 10^{-4} A/V^2$
γ	$0.0\sqrt{V}$	$0.2\sqrt{V}$
δ	$0.02V^{-1}$	$0.02V^{-1}$
Φ	$1.01V$	$1.28V$

MOSFET transistor - Level 2

- QPSIM constructor:

```
Mosfet::MosfetL2(SubCircuit* parent, char* name,
                 BaseNode& drain, BaseNode& gate,
                 BaseNode& source, BaseNode& bulk,
                 double ut0, double beta,
                 double gamma, double phi,
                 double delta);
```

- Parameter:

	enhancement type	depletion type
U_{T_0}	$2.0V$	$-2.43V$
β	$1.748 \cdot 10^{-3} A/V^2$	$5.35 \cdot 10^{-4} A/V^2$
γ	$0.35\sqrt{V}$	$0.2\sqrt{V}$
δ	$0.02V^{-1}$	$0.02V^{-1}$
Φ	$1.01V$	$1.28V$

Remark II.1. *The MOSFET level 1 and level 2 models differ in the way in which the bulk capacitances and the current gain are modelled. In the level 1 model, the bulk capacitances are assumed to be linear. In the level 2 model these capacitances are modelled as nonlinear capacitances.*

Bibliography

[1] Benchmark: NAND gate. www.math.hu-berlin.de/˜caren/nandgatter.ps.

[2] Daspk3.1. http://www.engineering.ucsb.edu/˜cse/software.html. Software program.

[3] Dassl. http://www.engineering.ucsb.edu/˜cse/software.html. Software program.

[4] Gelda. http://www.math.tu-berlin.de/numerik/mt/NumMat/Software/GELDA/. Software program.

[5] Genda. http://www.math.tu-berlin.de/numerik/mt/NumMat/Software/GELDA/. Software program.

[6] Geoms. http://www.math.tu-berlin.de/numerik/mt/NumMat/Software/GEOMS/. Software program.

[7] Matlab. http://www.mathworks.com/. Software program.

[8] Radau5. http://www.unige.ch/˜hairer/software.html. Software program.

[9] RadauP. http://www.unige.ch/˜hairer/software.html. Software program.

[10] Sundials. http://www.llnl.giv/CASC/sundials/. Software program.

[11] S. Bächle and F. Ebert. Element-based topological index reduction for differential-algebraic equations in circuit simulation. Technical Report Matheon 246, TU Berlin, Germany, 2005.

[12] S. Bächle and F. Ebert. Elementbasierende topologische Indexreduktion für differentiell-algebraische Gleichungen (DAE) in der Schaltungssimulation. Patentschrift DE 10 2005 023 145 A1, January 2005.

[13] A. Barrlund. Constrained least squares methods for linear timevarying DAE systems. *Numer. Math.*, 60:145–161, 1991.

[14] B. Bollobás. *Modern Graph Theory.* Springer, New York, 1998.

[15] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *The Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations.* SIAM Publications, Philadelphia, 1996.

[16] J. C. Butcher. Implicit Runge-Kutta processes. *Math. Comp.*, 18:50–64, 1964.

[17] J.C. Butcher. *Numerical methods for ordinary differential equations.* John Wiley & Sons Ltd., Chichester, 2003.

[18] S. L. Campbell. A general form for solvable linear time varying singular systems of differential equations. *SIAM J. Math. Anal.*, 18(4):1101–1115, 1987.

[19] S. L. Campbell. Least squares completions for nonlinear differential algebraic equations. *Numer. Math.*, 65:77–94, 1993.

[20] S. L. Campbell and C. W. Gear. The index of general nonlinear DAEs. *Numer. Math.*, 72(2):173–196, 1995.

[21] S. L. Campbell and E. Griepentrog. Solvability of general differential algebraic equations. *SIAM J. Sci. Comput.*, 16:257–270, 1995.

[22] G. G. Dahlquist. A special stability problem for linear multistep methods. *Nordisk Tidskr. Informations-Behandling*, 3:27–43, 1963.

[23] C. Desoer and E. Kuh. *Basic Circuit Theory.* McGraw-Hill, New York, 1969.

[24] D. Estévez Schwarz. Topological analysis for consistent initialization in circuit simulation. Technical Report 99–3, Institute of Mathematics, Humboldt University Berlin, 1999.

[25] D. Estévez Schwarz. A step-by-step approach to compute a consistent initialization for the MNA. *Int. J. Circ. Theor. Appl.*, 30(1):1–16, 2002.

[26] D. Estèvez-Schwarz and R. Lamour. The computation of consistent initial values for nonlinear index-2 differential-algebraic equations. *Numer. Alg.*, 26:49–75, 2001.

[27] D. Estévez Schwarz and C. Tischendorf. Structural analysis for electric circuits and consequences for MNA. Technical Report 98–21, Institute of Mathematics, Humboldt University Berlin, 1998.

[28] D. Estévez Schwarz and C. Tischendorf. Structural analysis for electric circuits and consequences for MNA. *Int. J. Circ. Theor. Appl.*, 28:131–162, 2000.

[29] D. Estévez Schwarz and C. Tischendorf. Mathematical problems in circuit simulation. *Math. Comput. Model. Dyn. Syst.*, 7(2):215–223, 2001.

[30] C. W. Gear. Differential-algebraic equation index transformation. *SIAM J. Sci. Statist. Comput.*, 9(1), 1988.

[31] C. W. Gear. Differential algebraic equations, indices, and integral algebraic equations. *SIAM J. Numer. Anal.*, 27(6):1527–1534, 1990.

[32] E. Griepentrog and R. März. *Differential-Algebraic Equations and Their Numerical Treatment.* Number 88 in Teubner-Texte zur Mathematik. Teubner Verlagsgesellschaft, Leipzig, 1986.

[33] M. Günther. *Ladungsorientierte Rosenbrock-Wanner-Methoden zur numerischen Simulation digitaler Schaltungen*, volume 168 of *Fortschritt-Berichte VDI, Reihe 20: Rechnerunterstützte Verfahren.* VDI-Verlag Düsseldorf, 1995.

[34] M. Günther and U. Feldmann. CAD-based electric-circuit modeling in industry. I. Mathematical structure and index of network equations. *Surv. Math. Ind.*, 8:97–129, 1999.

[35] E. Hairer, C. Lubich, and M. Roche. *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods.* Springer, Berlin, 1989.

[36] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I.* Springer, Berlin, 2nd edition, 1993.

[37] E. Hairer and G. Wanner. On stability of the BDF formulas. *SIAM J. Numer. Anal.*, 20(6):1206–1209, 1983.

[38] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II.* Springer, Berlin, 2nd edition, 1996.

[39] H. Häuser. *Gewöhnliche Differentialgleichungen.* B.G. Teubner Stuttgart, 1995.

[40] G. Reißig, W. S. Martinson, and P. I. Barton. Differential-algebraic equations of index 1 may have an arbitrarily high structural index. *SIAM J. Sci. Comput.*, 21(6):1987–1990, 2000.

[41] D. Jungnickel. *Graphs, Networks and Algorithms.* Springer, Berlin, 2nd edition, 2005.

[42] H. Khakzar, A. Mayer, and R. Oetinger. *Entwurf und Simulation von Halbleiterschaltungen mit SPICE.* expert Verlag, 1992.

[43] P. Kunkel and V. Mehrmann. Canonical forms for linear differential-algebraic equations with variable coefficients. *J. Comput. Appl. Math.*, 56:225–259, 1994.

[44] P. Kunkel and V. Mehrmann. Local and global invariants of linear differential-algebraic equations and their relations. *Electr. Trans. Num. Anal.*, 4:138–157, 1996.

[45] P. Kunkel and V. Mehrmann. A new class of discretization methods for the solution of linear differential algebraic equations with variable coefficients. *SIAM J. Numer. Anal.*, 33(5):1941–1961, 1996.

[46] P. Kunkel and V. Mehrmann. Regular solutions of nonlinear differential-algebraic equations and their numerical determination. *Numer. Math.*, 79:581–600, 1998.

[47] P. Kunkel and V. Mehrmann. Analysis of over- and underdetermined nonlinear differential-algebraic systems with application to nonlinear control problems. *Math. Control, Signals, Sys.*, 14:233–256, 2001.

[48] P. Kunkel and V. Mehrmann. Index reduction for differential-algebraic equations by minimal extension. *Z. Angew. Math. Mech.*, 84:579–597, 2004.

[49] P. Kunkel and V. Mehrmann. Characterization of classes of singular linear differential-algebraic equations. *Electron. J. Linear Algebra*, 13:359–386, 2005.

[50] P. Kunkel and V. Mehrmann. *Differential-Algebraic Equations. Analysis and Numerical Solution.* EMS Publishing House, Zürich, Switzerland, 2006.

[51] P. Kunkel and V. Mehrmann. Stability properties of differential-algebraic equations and spin-stabilized disrectizations. Technical Report MATHEON 356, TU Berlin, Germany, 2006.

[52] P. Kunkel, V. Mehrmann, W. Rath, and J. Weickert. A new software package for linear differential-algebraic equations. *SIAM J. Sci. Comput.*, 18:115–138, 1997.

[53] P. Kunkel, V. Mehrmann, M. Schmidt, I. Seufer, and A. Steinbrecher. Weak formulations of linear differential-algebraic systems. Technical Report 16-2006, Institute of Mathematics, TU Berlin, 2006.

[54] P. Kunkel, V. Mehrmann, and S. Seidel. A MATLAB toolbox for the numerical solution of differential-algebraic equations. Technical Report 16-2005, Institute of Mathematics, TU Berlin, 2005.

[55] P. Kunkel, V. Mehrmann, and I. Seufer. GENDA: A software package for the numerical solution of general nonlinear differential-algebraic equations. Technical Report 730-01, Institute of Mathematics, TU Berlin, 2003.

[56] R. Lamour. Index determination and calculation of consistent initial values for DAEs. *Computer & Math. w. Appl.*, 50(7):1125–1140, 2005.

[57] Ch. Liebchen. *Periodic Timetable Optimization in Public Transport.* PhD thesis, TU Berlin, Germany, 2006.

[58] R. März. Numerical methods for differential-algebraic equations. *Acta Numerica*, pages 141–198, 1992.

[59] R. März. Differential algebraic systems with properly stated leading term and the MNA equations. In *Modeling, Simulation, and Optimization of Integrated Circuits*, volume 146 of *International Series of Numerical Mathematics*, pages 135–152. Birkhäuser, 2001.

[60] R. März. Nonlinear differential-algebraic equations with properly formulated leading term. Technical Report 01-3, Institute of Mathematics, Humboldt University Berlin, 2001.

[61] R. März. Differential algebraic systems anew. *Appl. Numer. Math.*, 42(1-3):315–335, 2002.

[62] R. März. The index of linear differential algebraic equations with properly stated leading terms. *Result. Math.*, 3(4):308–338, 2002.

[63] R. März. Characterizing differential algebraic equations without the use of derivate arrays. *Computer & Math. w. Appl.*, 50(7):1141–1156, 2005.

[64] S. E. Mattsson and G. Söderlind. Index reduction in differential-algebraic equations using dummy derivatives. *SIAM J. Sci. Comput.*, 14(3):677–692, 1993.

[65] R. Melville. Design and development of a modular simulator, May 2004. Talk given at MATHEON Wokshop on Circuit Simulation at TU Berlin.

[66] U. Nowak and L. Niemann. A family of Newton codes for systems of highly nonlinear equations – algorithm, implementation, application. Technical Report TR 91-10, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1991.

[67] J. Ortega and W. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables.* SIAM Publications, Philadelphia, 2000.

[68] C. C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM J. Sci. Statist. Comput.*, 9(2):213–231, 1988.

[69] P.J. Plauger, A.A. Stepanov, M. Lee, and D.R. Musser. *The C++ Standard Template Library.* Prentice-Hall, New Jersey, 2000.

[70] K. Sakallah, Y. Yen, and S. Greenberg. A first-order charge conserving MOS capacitance model. *IEEE Trans. CAD*, 9(1):99–108, 1990.

[71] A. Schrijver. *Combinatorial Optimization*, volume A. Springer, Berlin, 2003.

[72] A. Steinbrecher. *Regularization and numerical simulation of nonlinear equations of motion of multibody systems in industrial applications.* PhD thesis, Institute of Mathematics, TU Berlin, 2006.

[73] J. Stoer and R. Burlisch. *Numerische Mathematik 2.* Springer, Berlin, 4th edition, 2000.

[74] B. Stroustrup. *The* C++ *Programming Language.* Addison-Wesley, Boston, 2nd edition, 2000.

[75] J. Vlach and K. Singhal. *Computer Methods for Circuit Analysis and Design.* Van Nostrand Reinhold, New York, 2nd edition, 1993.

[76] D. Ward and R. Dutton. A charge-oriented model for MOS transistor capacitances. *IEEE J. Solid-State Circuits*, 13(5):703–708, 1978.

[77] H. Whitney. On the abstract properties of linear dependence. *Am. J. Math.*, 57:509–533, 1935.

[78] P. Yang, B. Epler, and P. Chatterjee. An investigation of the charge conservation problem for MOSFET circuit simulation. *IEEE J. Solid-State Circuits*, 18(1):128–137, 1983.